LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHR- UND FORSCHUNGSEINHEIT FÜR
PROGRAMMIER- UND MODELLIERUNGSSPRACHEN

Institut für Informatik

Designing Image Labeling Games For More Informative Tags

Bartholomäus Steinmayr

Diplomarbeit

Beginn der Arbeit:	28. November 2010
Abgabe der Arbeit:	19. Mai 2011
Betreuer:	Prof. Dr. François Bry
	Christoph Wieser
	Fabian Kneißl

Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

München, den 19. Mai 2011 .....

Bartholomäus Steinmayr

Zusammenfassung

Die Erstellung textueller Beschreibungen von Bildern hat wichtige Anwendungen in Bilddatenbanken, barrierfreier Webgestaltung und computergestützer Bildverarbeitung. Die maschinelle Erstellung solcher Beschreibungen ist schwierig. „Games With A Purpose“ nutzen die Intelligenz menschlicher Spieler zur Beschreibung von Bildern. Bisherige Implementierungen sind höchst erfolgreich im Bezug auf die Anzahl der Spieler und der gesammelten Tags. Forschungsergebnisse deuten allerdings auf Probleme dieser Spiele hin, umfassende Bildbeschreibungen zu sammeln.

Wir stellen KARIDO, ein neuartiges Bildbeschreibungsspiel, vor. Ziel des Spiels ist das Sammeln differenzierter Tags. Diese Ausarbeitung beschreibt die Entwicklung und Implementierung zweier Versionen des Spiels, sowie deren Evaluierung in einer dreiwöchigen Testphase. Beide Versionen des Spiels werden miteinander sowie mit einem konventionellen Bildbeschreibungsspiel verglichen. Dazu wird die selbe Bilddatenbank verwendet. Die Ergebnisse dieser Untersuchung deuten auf eine starke Akzeptanz des Spiels hin. Eine detaillierte Evaluierung der Ergebnisse muss als zukünftige Arbeit verbleiben.

Abstract

Creating descriptive labels for pictures is an important task, with applications in image retrieval, Web accessibility and computer vision. Automatic creation of such labels is difficult. “Games With A Purpose” strive to create image descriptions by harnessing the intelligence of human players. Existing implementations are highly successful in terms of player count and number of collected labels. Research indicates that previous games struggle to create comprehensive tag sets containing both general and specific labels.

We propose KARIDO, a new image labeling game designed to collect more diverse tags. This thesis describes the design and implementation of two versions of the game, which are evaluated based on data collected during a three week trial deployment. The two versions of the game are compared with one another and an existing image labeling game using the same database of images. Results of this evaluation indicate strong player acceptance of the game. A detailed evaluation of tag quality is left as future work.

Acknowledgements

While the cover of this thesis only names one author, a diploma thesis is never the product of one sole person. I would like to express my gratitude to all those who have enabled me to complete this thesis.

First on this list are Christoph Wieser and Fabian Kneißl, who not only helped me with words and deeds whenever the SEAM Web framework got jammed, but also ensured that I felt at home at the institute from the very first day. My gratitude for this friendly welcome also belongs to Prof. Dr. François Bry, all members of the teaching and research unit for *Programming and Modelling Languages* as well as all participants of the Artigo project. Furthermore, I would like to thank Prof. Bry for his enthusiasm for this thesis and his support for its completion.

Last, but most certainly not least, credit is due to my friends and family, who in the last six months made sure that my life was not spent entirely in front of computer screens.

Bartl Steinmayr

Contents

1. Introduction

In the last few decades, computers have turned from academic curiosities into industrial machines and finally into ubiquitous devices that penetrate nearly all aspects of modern life. Many day-to-day things we take for granted would simply be impossible without the immense computational capabilities of today's computers. There are airplanes that would not be able to sustain flight [4], if not for the constant meticulous corrections of highly complex control systems. Computers can combine and analyze data from thousands of measurement stations all over the world to create increasingly reliable weather forecasts [23]. Many trains today even run without human drivers [34].

This list could be extended to fill a thesis of its own. Suffice it to say that overall computers are faster, more accurate and less error-prone than human intuitivity in solving a wide variety of problems. However, the more difficult and diverse these solved problems are, the more apparent the unsolved ones become: Understanding the meaning of a spoken sentence, distinguishing a cat from a dog or describing the contents of a picture is no challenge at all for an average five-year-old, but is currently challenging for even the most advanced computer systems.

One way to approach these somewhat surprising deficiencies is to try and find ways in which computers can "think" in a more human way. Examples of this include neuronal networks and other machine learning approaches. Similarly, one can try to transform these problems to be more accessible for computers and find better algorithms.

A different approach – the one taken in this work – is to simply accept the different "skill sets" of humans and computers. The goal is then to try and find ways to harness the special capabilities of the human mind and combine them with the skills of machines.

1.1. Early Experiments in Human Computation

Von Ahn calls this process of harnessing the users' intelligence *Human Computation* [24]. A large number of alternative terms are used to describe collaborative problem-solving, including *Crowdsourcing*, *Peer Production*, *Collective Intelligence* and *Crowd Wisdom* [6]. The first experiments in utilizing human intelligence presented tasks to their users directly. Only later these tasks were turned into the games which are the focus of this thesis. Nevertheless, the early systems form the base from which these games evolved.

1.1.1. Open Mind Initiative

One of the first attempts to use the mental capabilities of laypersons to solve scientific problems was the Open Mind Initiative [22]. The Open Mind Initiative was founded by Stork to improve intelligent systems such as text- or pattern recognition. Most of these systems heavily rely on large sets of training data, the acquisition of which can be difficult – after all, one cannot use computers to create them.

Stork proposes a three-tiered structure for the creation of intelligent systems. Firstly, domain experts provide the fundamental models and algorithms. The creation of training

data for these systems is too labor intensive to be performed by a small group of specialists. Stork therefore propose to allow large groups of laypersons to provide the necessary training data as the second tier. Finally, the Open Mind Initiative provides a framework for recruiting and motivating volunteers and collecting the data using the Web.

Since training datasets should usually be as large as possible, it is desirable to recruit as many volunteers as possible. Stork names a number of incentives for motivating participants. These incentives include altruism and public recognition. Stork furthermore proposes to design the interfaces of the data collection systems as games. The experience of playing these games should be pleasurable, thus motivating users to participate. This is the first mention of what would later be called *Games With A Purpose* (see Section 2.1) by von Ahn. However, Stork and other members of the Open Mind Initiative did not focus on the creation of games in their further research. The Open Mind Initiative will therefore not be discussed in more detail in this work.

1.1.2. CAPTCHA

An interesting way to take advantage of the differences between the calculations of machines and human thinking was formalized by von Ahn et al. [27]. In many instances, it is desirable to allow only human users access to a given system. Examples include communication tools such as Web forums or email accounts, which could be misused to distribute unwanted advertisements. Another example is an online poll, which could easily be manipulated if automated voting were possible.

Von Ahn et al. devised a simple yet effective way to assert that a user is human: a *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA). The system relies on the assumption that there are some challenges which computers currently cannot solve. However, in some cases, computers can automatically generate such challenges. For example, it is trivial to create a program that transforms a given text into an image and distorts it, but it is challenging to create a program that reliably reverses this process.

The potential users of a system (whether they are humans or computers) are therefore presented with such a generated problem and are challenged to reverse the generation process. The solution to the challenge is known to the system, because it has used the solution to generate the problem in the first place. Thus, the system can easily verify whether an entered solution is correct. Therefore – since only a human can correctly solve the challenge – legitimate users can be identified as such.

The test is public, in that the data and algorithms used to create the challenges are publicly accessible. If this were not the case, faulty designs could lead to easily breakable challenges. Consider, for example, a database of images, each depicting a given subject and a manually created list of these subjects. Users could then be shown pictures and be asked to describe the content. This is currently not possible for a computer. However, if the database were not sufficiently large, an attacker could manually solve the challenges. In the future, a computer could then compare the image of a challenge with this manually created database of solutions. Incontrast, as CAPTCHAs do not use secret databases, the only way to circumvent the Turing test is to solve the underlying challenge. This creates a

win-win situation for the creator of a CAPTCHA: Either the challenge remains unsolved and the CAPTCHA remains secure or a currently difficult problem of artificial intelligence has been solved. Von Ahn et al. call this "lazy cryptographers doing AI".

The authors later recognized that human cognitive resources are wasted by using randomly generated tests. They therefore enhanced the system to what they call reCAPTCHA [31]. Instead of using random words, the system creates challenges with words taken from scanned books. Recognizing the text in scanned documents is still difficult despite the advances in OCR[1] technology, especially when the quality of the print or paper is poor. ReCAPTCHA selects any words which were not recognized identically by several different text recognition programs. The images are furthermore distorted to make automatic recognition even more difficult.

The problem with this approach is that the solution to a challenge is not known to the system a priori. Therefore, the system cannot decide on its own whether a given answer is correct. Von Ahn et al. solve this by showing the user two words, one which is already known and one which has yet to be deciphered. The answer of the user is only accepted if the known word is correctly transcribed. Since the user does not know which is the known word, she can only try to enter both words correctly. If several users have given the same solution to an unknown word, it is considered transcribed correctly and added to the list of known words.

Using this approach, it is possible to identify human users and at the same time accurately decipher scanned books. Even though the solution of a CAPTCHA usually only requires seconds of human cognition, the vast number of users constitutes a large computational resource: As of 2008, reCAPTCHA users collaboratively transcribed 160 books per day, with an accuracy far exceeding automated systems.

1.2. Image Labeling Games

After the creation of the CAPTCHA, von Ahn recognized another important fact. Aside from the time wasted through necessary evils like CAPTCHAs, people love to waste time voluntarily. Von Ahn estimates [24] that several billion human-hours are spent each year playing solitaire[2]. If properly channeled, this energy could be used to tackle previously unsolved large scale computation problems. Von Ahn proposes to do this by creating so called *Games With A Purpose* (for better readability, the abbreviations "GWAP" and "GWAPs" will be used to distinguish "Game With A Purpose" and "Games With A Purpose" respectively). A more detailed explanation of the fundamental mechanisms of GWAPs can be found in Section 2.1.

The first application of a GWAP demonstrated by von Ahn and Dabbish was image labeling [25]. Image labeling is the process of creating textual meta-data describing the content of an image. These meta-data are usually divided into single attributes called *labels* or *tags* (both terms will be used synonymously from now on). These attributes can include both abstract information, such as the type of the image (e.g., "photograph", "drawing"

[1] Optical Character Recognition describes the process of extracting the textual content from the writing contained in an image.

[2] Solitair is a popular computer card game included with the Windows operating system.

or "painting"), as well as concrete descriptions of the contents depicted in it (e.g., "small dog" or "tree"). Automating this process is a highly researched subject in the field of machine learning, but current automatic solutions cannot rival the accuracy of human-created descriptions.

Manually annotating images can be feasible for a private collection of photographs, but current public image databases often contain millions of different images. Paying workers to manually label these images is usually not financially feasible and relying solely on volunteers raises the issue of how to motivate the participants. To solve this problem, von Ahn and Dabbish created the so-called ESP Game (see Section 2.2.1), in which two players collaboratively create textual descriptions of images.

1.2.1. Applications of Image Labeling

There are several reasons for creating accurate textual descriptions of images. Arguably the most important application for image labeling is image retrieval. Selecting stored images in a systematic fashion is one of the fundamental functions of any image database. As a special case, the Web can be considered a very large image database. Usually, queries to such a database refer to the content of the stored images. For example, a user might want to access an image depicting a specific object. The following paragraph lists a number of image retrieval methods and explains why image labeling is important.

One possibility for handling such requests is *query by example*. The user provides a sample image and the database returns images that are similar to this image by some metric. An implementation of this technique is the TinEye image search engine [tin]. One issue of this method is that the user has no method for specifying which trait of the sample image is important (images contain colors, shapes and specific objects, all of which could be, but do not need to be important in a query). Furthermore, this approach requires that the user is already in possession of a suitable example image. Lastly, query by example does not allow searching for generic classes of images (for example, images containing any kind of "building").

Another approach for creating visual queries is through *sketches*. Users can create simple drawings and the image database should return objects that match this sketch in shape and/or color. For this approach to return accurate results, users need to be able to create detailed and accurate drawings. Existing implementations of this query approach (see the works of Eitz et al., Chen et al., Gavilan et al.) are designed for casual applications such as creating collages from personal images. This suggests that sketching alone is not sufficient for accurate selection of images. Chen et al. circumvent this problem, by combining querying by shape with textual queries [3].

Using textual queries is arguably the most common image retrieval technique. Many image database systems (e.g., Google Images [goob] or Prometheus [pro]) primarily rely on textual descriptions for querying images. This means that each image is assigned a symbolic description against which any entered query is matched. Since the content of the images does not need to be considered while processing the queries, database requests can be performed using fast and proven algorithms for document retrieval. Furthermore, users can describe the image they are looking for without having to provide examples or needing

to be able to draw. However, the results of textual queries will only be accurate and reliable if the descriptions of the images reflect their actual content.

In some cases, image descriptions can be created automatically without analyzing the content of the images. On the Web, images are often accompanied by text. By considering spatial proximity between images and texts, potentially relevant keywords for images can be extracted from Web-sites. Nevertheless, this extraction is error-prone and fails completely for images without text. Thus, image labeling is highly important for image retrieval.

In turn, an important application for image retrieval is the filtering of content. In some cases, restricting the access of users to certain materials is desirable. For example, parents might want to allow their children access to the Web, but protect them from pornography and violence. Appropriately labeled images could solve this issue reliably.

1.2.2. Goals of Image Labeling

As the arguably most important application, image retrieval defines the desirable properties of image labeling. Therefore, the labels for an image should reflect the structure of common queries. Weber et al. state that users formulating queries to an image database are more likely to use generic terms than specific ones [32]. Therefore, the existence of a sufficient number of generic labels for each image is crucial.

However, Weber et al. also note that the Web shows an important problem that has to be considered. On the Web, a large number of generically labeled images already exist – a search for "car" on Bing (formerly Live) Image Search [bin] returns 150 million results. In contrast, queries using highly specific labels such as a car make and color return too few or incorrect results. Therefore, it is also important that images are labeled with as many specific tags as possible.

In combination, the above statements lead to the following conclusion: The best results in image retrieval will be achieved if images are labeled with a comprehensive set of labels, including both generic and specific ones.

1.3. Motivation for Creating a New Game

Tag Diversity. As discussed above, the goal of image labeling is to create a comprehensive set of descriptions for any given image.

The basic mechanics of existing games are designed to enforce correct labels, but not necessarily comprehensive ones. Probably the most important example for this is the ESP game. It is important because the problem is prominent in the ESP Game and because it is one of the few implementations that are used productive systems. As described in Section 2.2.1, the ESP game pairs two random players who have to agree on a description of an image. Thus, each player does not know anything about her partner and thus has to assume she is not an expert in a given domain. Therefore, players are more likely to achieve a match if they enter a generic term as opposed to a specific ones. This has been proven using *game theory* by Jain and Parkes [13] (see Section 2.4.1). The basic game design of the ESP game would thus yield specific descriptions only with a small likelihood.

Current Solution. Von Ahn and Dabbish noticed the issue of tag diversity. As a solution, they propose the use of so called *taboo words*. Any tag entered for a given image is added to a list (such a list is kept for each image). If a tag is already on the list, the number of times it has been assigned is counted. Once this number of occurrences of a tag exceeds a given threshold, the tag is added to the list of taboo words for the given image. This list is shown to players in later rounds and the players cannot enter the displayed labels any longer. Taboo words are meant to force players to use different and eventually more specific terms.

Weber et al. performed an extensive evaluation [32] of the results of a version of the ESP Game used by Google [gooa]. Since the results of this game are not publicly available, the authors extracted the displayed taboo words, which reflect the labels assigned to an image. Despite their restriction to taboo words, Weber et al. found a large number of redundant and generic labels in the extracted data. Furthermore, many tags where highly correlated. As an example, the authors mention that 68% of all images labeled with "clouds" were also labeled with "sky". Weber et al. therefore argue that this kind of data does not necessarily need to be created by human players. To prove their point, they implemented a software that successfully plays the ESP Game. This is somewhat paradoxical, since the main objective of the game (i.e., labeling images) cannot yet be achieved reliably by computers.

The software created by Weber et al. disregards the visual content of the images. Instead, it predicts which tags its partner is likely to assign by analyzing the taboo words. To achieve this, the authors use a Bayesian model of tag probabilities, incrementally trained on the data acquired while playing the ESP Game. The software played over 400 games and achieved a match in about 80% of these games. This means that the tags entered by human players are highly predictable given the taboo words. Thus, human players add little information to the existing tags and indeed probably try to find synonyms for the given taboo words.

In addition to the creation of this software, Weber et al. propose other solutions for the problem of tag diversity. They suggest a scoring scheme in which players are rewarded for assigning more informative tags. Such scores can be computed using the previously established likelihood of a term, given the current taboo words. Nevertheless, Weber et al. mention that it remains questionable whether players actually change their behavior according to the scoring.

Another proposed solution is to hide the taboo words. To avoid players simply ignoring the taboo and proceeding as usual, there needs to be a punishment for triggering a taboo word. Weber et al. propose to end the round immediately with a score of zero if the players agree on a blocked term. This would encourage players to use more specific terms, as general ones are more likely to already be present. Furthermore, players would not be biased by the terms already displayed.

However, failing a round for a reason that is not predictable is likely to frustrate players. Furthermore, because of the severity of the punishment, players are forced to be very careful in their choice of tags, making a positive match even more unlikely. This will further increase frustration and lead to a decreasing number of players. A different solution for increasing the tag diversity in image labeling games is proposed in Section 3.

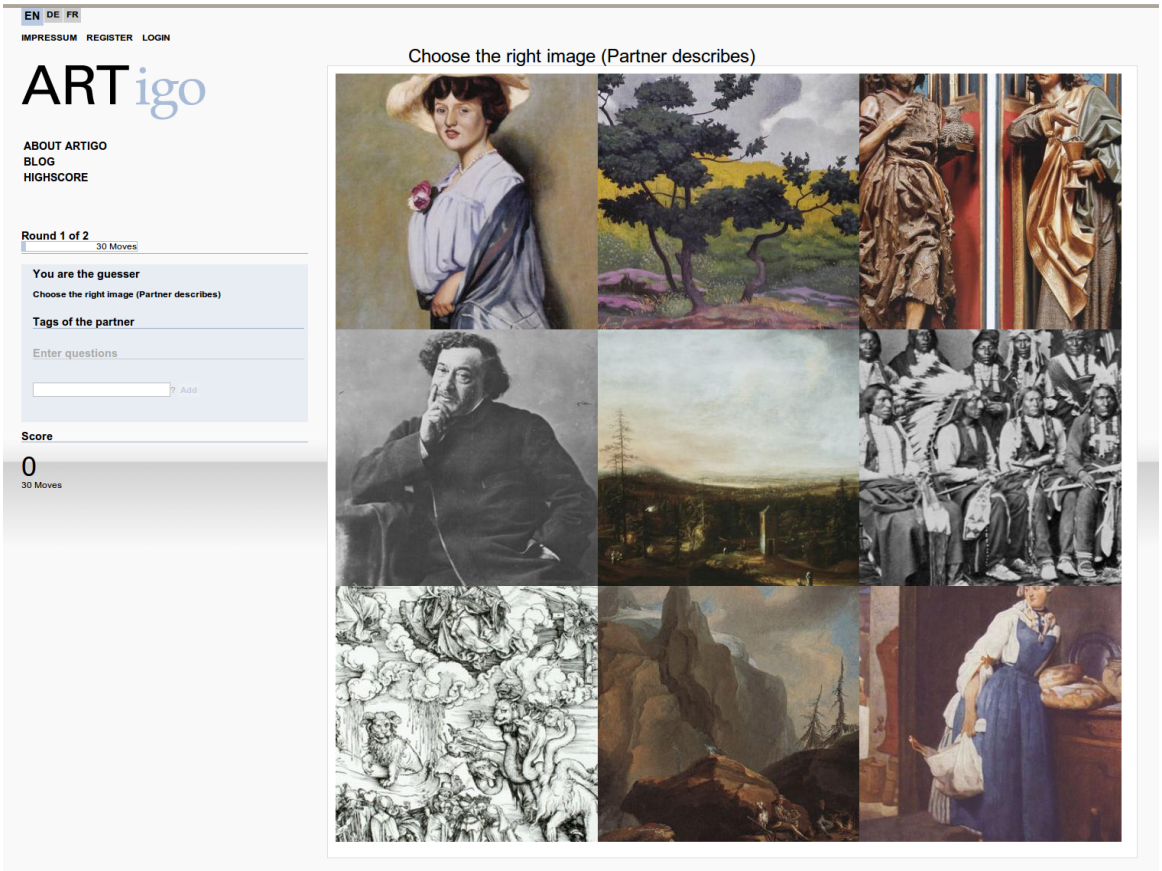


Figure 1: Screenshot of Karido

1.4. Proposed New Karido Game and Related Approaches

Aside from the use of taboo words, a number of image labeling games have been created using various strategies for ensuring tag diversity. A discussion of these games can be found in Section 2.

Like most of the games mentioned above, the proposed game is designed to be collaboratively played by two players. A grid of images is shown to both players (see Figure 1). The order of the images is randomized for each player. In the grid of the first player, one image is highlighted. Their goal is to describe this image to their partner. The second player must then select the right image from their grid. By selecting the images in the grid by tag similarity, it is possible to force players to contribute new information. The proposed game is called Karido, as suggested by Gerhard Schön. This name is meant to evoke associations of the regular grid (french "carré"), while also highlighting its relation to Artigo. Previously, the game was called Memoria, in reference to the popular card game Memory [Hasa].

The basic mechanism of Karido is quite similar to that of Verbosity [29], created by von Ahn et al. (see Section 2.3.1). However, Verbosity is not an image labeling game and lacks the concept of using input similarity (see Section 3.1.1) to increase tag diversity.

Scope of this Thesis. Following the arguments mentioned above, image labeling games should be designed with both correctness and diversity of tags as their fundamental goals. The aim of this thesis is to present and evaluate a new design for an image labeling game.

Section 1 of this thesis is this introduction. Section 2 contains an overview of related work and current implementations of image labeling games on which my work is based. This Section furthermore contains several concepts introduced by von Ahn which can be applied to arbitrary Games With A Purpose. Section 3 describes the design of the proposed new game and the expected advantages that have guided its creation. In Section 4 an overview of the implementation of a prototype of the game is given. This part of the thesis furthermore includes an introduction to the framework created for the game, as well as special challenges faced during the implementation and their respective solutions. Section 5 summarizes the empirical results gathered from the prototype. Finally, the conclusions drawn from these results and possible future work are outlined.

2. Related Work and State of the Art

The following Section contains a short description of the history of *Games With A Purpose* and an overview of the state of the art of GWAPs. The discussed related works include both games used image labeling as well as games designed for other purposes.

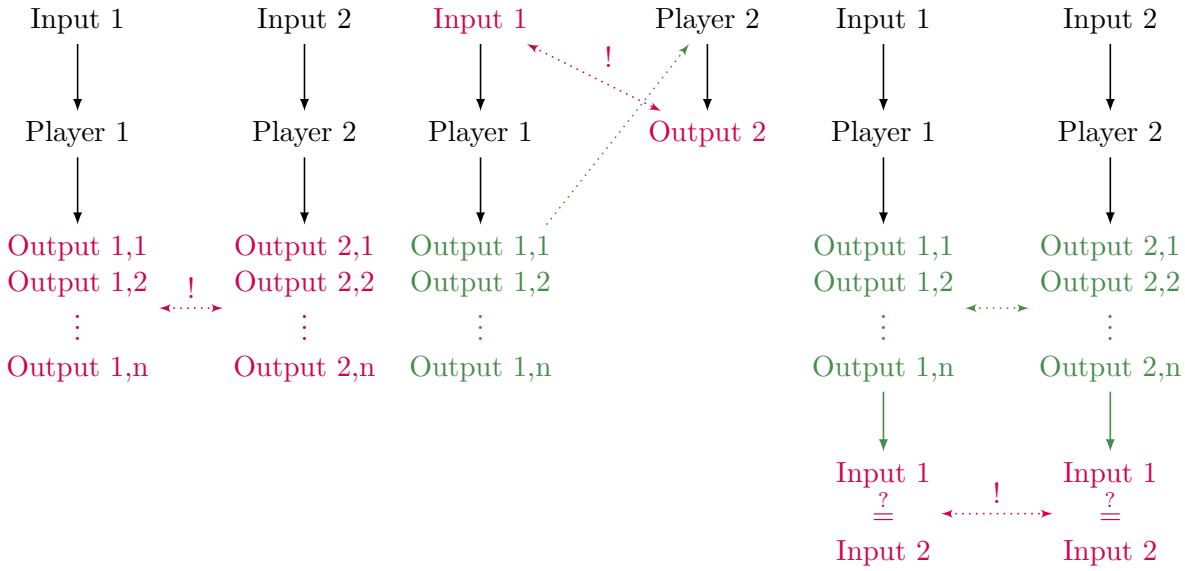
2.1. The Evolution of Games With A Purpose

As described in the previous Section, GWAPs evolved from data gathering systems. Their main goal is to motivate players to contribute time and effort for solving scientific problems. These games should thus be designed to be fun to play to incite as many users as possible. At the same time, GWAPs must perform some kind of calculation while being played. This *purpose* of the game does necessarily have to be communicated to the player, but is the essential characteristic that separates GWAPs from games that only aim to entertain.

Von Ahn proposes to treat these games like computer algorithms [24]. Instead of being executed by computer systems, these algorithms are being "run" on the highly distributed minds of the players. Other than that, the same considerations as for any other algorithm apply. The game should be correct, meaning that irrespective of the individual players' intentions, the relationship between input and output of the game must conform to a given specification. Von Ahn suggests to achieve this by making correctness of the output the winning condition for a given game. At first, this sounds like a paradox. The problems tackled by Games With A Purpose are those to which the solution cannot be computed. This usually means that the correctness of a solution cannot be verified automatically either.

However, in many instances the players themselves can verify a solution. In all cases, the players are required to be independent. Depending on the design of the game and the types of information involved, the players may also need to be unable to communicate outside of the game. If these preconditions apply, there are several possibilities of enforcing correct output. Von Ahn proposes two general verification schemes [24] and later added a third one [26].

- In *Output-agreement* games, all players involved assume the same role. Their output will be accepted only if both players independently agree on it. (This scheme is illustrated in Figure 2a. Section 2.2.1 on the ESP GAME describes a game of this type.)
- In *Inversion-problem* games, the players assume different roles. The first player transforms a given input into an intermediate output. This output is passed to the second player. The second player's task is to reconstruct the first player's input. If she succeeds, the intermediate output is accepted. (This scheme is illustrated in Figure 2b. The KARIDO game described in Section 3 of this thesis uses this verification scheme.)
- In *Input-agreement* games, both players transform a given input into an intermediate output. In contrast to inversion-problem games, both players can see the intermediate output of their partner. They then try to agree on whether their respective inputs are identical. (This scheme is illustrated in Figure 2c. Section 2.3.3 on TAGATUNE describes a game of this type.)



(a) Output-agreement (b) Inversion-problem (c) Input-agreement

Figure 2: Output verification schemes for GWAPs
Any data shared by both player is green; Data that is used for verification and has to be identical for the players to succeed is magenta

Despite the diversity of existing Games With A Purpose, almost all of them use variations of the described methods. The correctness of the output cannot be guaranteed using any of the described mechanisms. For example, both players could make a mistake and still agree on the erroneous output. Alternatively, malicious users could circumvent the communication barrier and deliberately create false data (see Section 2.2.1 for further exploitation scenarios and appropriate countermeasures). These and other scenarios which lead to incorrect data cannot be avoided entirely. However, if the aforementioned preconditions apply and a verification method is used, the game design can be adjusted to return correct solutions with an arbitrarily high probability.

Aside from being correct as described above, an algorithm should also be efficient. For computer algorithms, this means that the time and memory required for finding a solution to a given problem should be as low as possible. This requirement is applicable to the implementation of a game, but not necessarily to the game itself. To evaluate the efficiency of Games With A Purpose, von Ahn proposes a combination of two measures [24]:

- The *throughput* of a game measures the average number of problems solved per player in a given time span. This is an indication of how effective players are when they are playing the game. However, a game with a high throughput can still be inefficient if no one is willing to play it. The enjoyability of a game is a very important aspect of its quality and is not captured by the measure of throughput.
- The *Average Lifetime Play* (ALP) of a game is defined as the overall average time one single player spends playing the game. Since players are less likely to quit playing a

fun game, this measure is indicative of the enjoyability of a game.

The combination of both measures yields the *expected contribution* of a game. This is the average number of problem instances solved by an individual player of the game. Therefore, given a specific number of users, the game with the highest expected contribution will return the largest number of solved problem instances.

Since the creation of the ESP Game by von Ahn and Dabbish, a large number of Games With A Purpose have been designed, aiming to solve a wide variety of problems. The following Section of this thesis describes a selection of GWAPs, both for image labeling as well as other purposes. Figure 3 contains an overview of the games discussed below.

Verification \ Game type	Image Labeling	Other
Output-agreement	ESP Game KissKissBan Matchin Picture This Magic Bullet	People Watcher OntoGame
Inversion-problem	Peekaboom Phetch Karido	Verbosity
Input-agreement		TagATune Intentions
Hybrid	PhotoSlap	Foldit CityExplorer Page Hunt

Figure 3: Games With A Purpose presented in this thesis

2.2. Image Labeling Games

Although image labeling appears to be a relatively narrow field of application for GWAPs, a number of different image labeling games have been designed. The earliest and most popular image labeling game is the ESP Game (see next Section) developed by von Ahn and Dabbish. Several other games have been developed (by the group around von Ahn as well as other groups) as extensions or improvements to the ESP Game. Additionally, there are a number of games that use radically different game mechanics. The following Section gives an overview of these existing implementations and their respective gameplay.

2.2.1. ESP Game

The ESP Game[3], created by von Ahn and Dabbish [25] is not only the first image labeling game, but also the first Game With A Purpose. Like all image labeling games, the functional

[3]The name ESP Game stems from *Extra-Sensory Perception*, a concept that describes the communication of information between humans without relying on senses, but solely on the mind.

goal of this game is to create descriptions of images in natural language.

Basic Gameplay. As in most Games With A Purpose, two players are randomly paired without the possibility to communicate directly. From now on, a player that has been assigned a partner in this way is refered to as *matched*. A picture is then shown to each of them. This picture is identical for both players and is randomly selected from a large database of images. While the players cannot communicate directly, they can enter arbitrary terms into the game. If both players enter the same term, they are awarded with a score bonus and a new round is initiated. The process is then repeated with a new image. An overall time limit is imposed to create a winning condition for the game: To collect as many points as possible in the given amount of time. The ESP Game thus is an example of an output-agreement game.

Although this is not stated in the game, players recognize that the best strategy to match with their partner is to use the displayed image as reference[4]. Therefore, successful players of the game create accurate descriptions of the images while playing the game.

The ESP Game met with tremendous success: In the first four months alone, over 13000 people played the game, more than 80% of them multiple times. Collaboratively, these players have created almost 1.3 million labels. This illustrates the enormous potential of Games With A Purpose to incite people to contribute time and effort to solving scientific problems and is a breakthrough in several aspects: Manual image labeling used to be a tedious process and could not be automated. It has now been transformed into a game that people are willing to play voluntarily for their own entertainment. Furthermore, because two players have to match on a description, the tags are inherently validated. A simple, manual labeling system without validation mechanisms could be manipulated very easily. Even if a majority voting scheme were applied, a larger group of attackers could still create false data. However, unless a group of attackers constituted a significant percentage of the users of the ESP Game, creation of false data is not possible in this game.

Replay Rounds. All Games With A Purpose that require a specific number $n \geq 2$ of players face a problem: At any given time, there may be up to $n-1$ players who cannot play until other players join the game. Especially when a game is new and/or not very popular, there might not be a steady stream of players joining the game. Thus, players might have to wait a very long time before they can play. This is likely to deter them from returning to the game and will prohibit that the game ever becomes popular. Players might furthermore be frustrated if their partners suddenly leave a match (or are disconnected because of technical reasons). Therefore, it is important for any GWAP to have a way of handling unmatched players.

In the ESP Game, two players are paired per game session. Therefore, at most one player can be unpaired. Von Ahn and Dabbish propose to handle this single player by replaying previous rounds. Any terms a player enters for a given image are recorded, along with the time it took them to enter these terms. This makes it possible to replay entire rounds

[4]This is not the case if further communication between the players is possible. Several measures were taken to avoid this, see Section 2.2.1.

exactly like they were previously played. Since the players cannot communicate directly, the simulated player does not have to react to the input of the real player. Thus, no difference in the behavior is noticeable for the human player.

In addition to providing the possibility of playing alone, the replayed rounds also create valid data. As the players cannot communicate directly, it is not important whether two players actually play at the same time or a round is recorded and replayed later. Therefore, when a human player produces a match in a replayed round, this match is as valid as if it were produced by two players playing at the same time. Replaying a round can therefore be considered a validation of previously collected labels. Furthermore, since the input of the human player is also recorded, new terms are added to the database and can be validated later.

The reliability of the output could be decreased if a player were to validate a round that she previously played herself. This can be avoided by forcing players to register and selecting only rounds from different players for replay. Furthermore, since the players of the ESP Game number in the thousands, the likelihood of a player replaying their own round is quite low.

One issue exists in the early stages of a game. When few rounds have been recorded, there is a relatively high probability that a player will be given the same round twice. Especially if the number of players remains low, this can become noticeable for the players. von Ahn and Dabbish do not discuss this issue, possibly because of the rapid success of the ESP Game.

Taboo Words and Label Threshold. To decide whether a label should be accepted for an image, von Ahn and Dabbish use a threshold value. The rationale is that a term on which a single pair of players matched could still be incorrect. Players could have manipulated the game (see next paragraph) or matched on a wrong term by chance. The ESP Game accepts a tag only after it has been applied at least n times. By adjusting n, the size of the collected tag set can be traded off against the reliability of the collected tags. The ESP Game uses $n = 1$ thus maximizing the number of collected tags.

As described in Section 1.3, all accepted terms are added to a list of taboo words for the image. These terms (or any terms containing them) can then no longer be applied to the given image. This is meant to increase the diversity of the tags applied to an image.

Measures against Cheating. Cheating is an important problem to consider when developing any multi-player game, but especially when designing Games With A Purpose. Players might cheat to acquire a high score more easily or with the explicit malicious goal of manipulating the output of the game. In both cases, the reliability of the output (in case of the ESP Game, the created labels) is dramatically decreased.

In the ESP Game, most straight-forward way to cheat is for players to communicate outside the game. This includes the following scenarios:

- Two players that are matched can communicate, for example by electronic means or because they are in the same room or even at the same computer.

- A single player operates several instances of the game and is matched with herself.
- A group of players have agreed on a common strategy before playing the game.

The most important strategy against these attacks is random pairing. This means that players must not be able to manipulate with whom they are paired (for example by joining the game at the same time). To ensure this, players joining the game are placed in a queue and are paired only if a sufficient number of players is waiting. Furthermore, players are paired only if they possess different IP addresses, making it difficult to operate several instances from a single computer.

Further measures are taken to prevent global agreement strategies. Such strategies are relatively unlikely to succeed in the first place. The ESP Game is usually played by hundreds or thousands of players at a time. Therefore, unless the percentage of players participating in the strategy that was agreed on is very high, it is quite unlikely that two players following the strategy are paired.

Nevertheless, von Ahn and Dabbish propose further measures to prevent global strategies. The first proposal is the extension of taboo words. Each term entered in a single round of the game would be taboo for the entire remainder of the game session. This would prevent simple agreement schemes in which each player assigns the same tag for each image. However, cheaters could resort to more complicated schemes, such as entering "one" for the first image, "two" for the second and so on. Von Ahn and Dabbish claim that it is possible to detect such strategies by the decrease in the time it takes two players to agree. As a solution, players could then be given an increased number of replayed rounds, in which these strategies do not lead to matches. Malicious players should then be discouraged quickly.

2.2.2. Peekaboom

Peekaboom was created by von Ahn et al. to gain more detailed descriptions of images [30]. Whereas the ESP Game returns only descriptions for the entire image, Peekaboom is designed to return the regions of the objects depicted in an image.

Basic Gameplay. Like in most other Games With A Purpose, Peekaboom pairs two random players for each game session. However, Peekaboom is an inversion-problem game. The ESP Game is symmetric, meaning both players perform the same types of actions in any given game round. In contrast, the players of Peekaboom assume two different roles. An image is shown to the first player, along with a description of an object in the image. These descriptions could, for example, be generated using the ESP Game. The first player can successively reveal parts of the image to their partner. Von Ahn et al. call this *Boom*. The second player can only see the areas of the image that were revealed to her. This is called *Peek*. Her goal is to guess the term that was shown to the Boom player. If she achieves this goal, both players are awarded with a score bonus and a new round is initiated. Again, an overall time limit is enforced.

Since the players cannot communicate, the Boom player must strive to reveal only the parts of the image that contain the named object. If she reveals parts of the image that are irrelevant to the given term, guessing this term precisely becomes difficult. Thus, the Peek

player will only be able to guess the right term if the Boom player reveals only the parts of the image related to the term. By combining the data of several players, the location and area of the object in an image can be determined with high accuracy.

Replay Rounds. Like other Games With A Purpose, PEEKABOOM requires the possibility of a single person playing alone. Because of the asymmetric nature of the game, this is more difficult to implement than for the ESP GAME. Simulating the Boom player is analogous to simulating a player in the ESP GAME: Entire game rounds are recorded and replayed unchanged.

Simulating the Peek player is more difficult, since this player has to react to the input of their partner. Since the correct answer of the Peek player is known to the game (because it handles both the Boom and the Peek player), simulating the right answer is trivial. However, the simulated player should not answer correctly if the wrong image region was revealed. Since the data entered by a Boom player is validated by later replaying the recorded round, no false data can be introduced into the system. This would remain true even if a simulated player always guessed the right term. However, players would likely figure out this behavior and no longer introduce correct data into the system.

In the long run, it is therefore important to maintain the illusion that a player is matched with another human. To achieve this, von Ahn et al. rely on the data that was previously verified by human players. By comparing the input of a human player with the existing data, wrong input can be detected. If this is the case, the simulated player returns incorrect answers.

2.2.3. PhotoSlap

PHOTOSLAP, created by Ho et al., is inspired by an existing card game known as SNAP [11]. It is not an image labeling game, but is instead designed to detect images which depict the same person. Because of the similarity of this application to image labeling, it is listed alongside the image labeling games.

Basic Gameplay. In comparison to other Games With A Purpose, the rules of PHOTOSLAP are relatively complex. The game is designed to be played by at least three users, with – in principle – no upper limit for the number of players. The implementation by Ho et al. is designed for four players. A set of images is selected randomly for each game round. These images are shown to the players before the actual game begins. Each player can set a number of *Traps* on arbitrary pairs of images. When all players are finished, the actual game round begins. During the game, the images are revealed successively in a pseudo-random order determined by the game. At any time, players can *Slap* if they believe that the last two images depicted the same person. If no other player *Objects*, the player who Slapped is awarded a score bonus. Otherwise, the player who Objected gains points and the player who Slapped loses points, unless a Trap was set on the pair of images shown. If an Objection falls into a Trap, the player who Objected loses points and the player who set the trap gains points. Finally, a player looses points if no other player Slaps on a trap they

have set and gains points if this is the case. To avoid dishonest behavior, a player cannot slap on a trap they have placed themselves.

The action of Slapping marks two photos as depicting the same person. Objections are introduced to verify this data: If a player cheats by Slapping on images that do not depict the same person, the other players can revoke her action through Objecting. The settings of Traps in turn controls Objections and prohibits that players Object a valid Slapping. If no Traps were set on valid pairs of images, malicious players could Object to all Slappings without consequences, defeating the purpose of the game. Finally, the deduction of points for Traps that were not slapped on, avoids setting of invalid Traps.

Analysis. The convention of the game is to Slap on images which depict the same subject. Given this convention, Ho et al. prove through game-theoretic analysis that the winning strategy is to indeed Slap and put Traps on images which depict the same subject and Object to Slaps on images which do not. However, the game would work identically for different conventions, for example Slapping on images which both depict a person wearing a tie. Therefore, players have to be informed of this convention. If this is the case, rational players will follow the convention and produce correct results.

2.2.4. KissKissBan

KissKissBan is an image labeling game designed by Ho et al. [12]. It extends the basic mechanism of the ESP Game to avoid the issues mentioned in Section 1.3 (i.e. tag diversity).

Basic Gameplay. The fundamental difference between KissKissBan and the ESP Game is the introduction of a third player and a competitive element in KissKissBan. The first two players are called *Couple* and try to achieve the same goal as in the ESP Game (which is called *Kiss* by Ho et al.). An image is shown to both players, who try to enter matching terms, without communicating, before a time limit of 30 seconds runs out. A minor difference to the ESP Game is that the time limit is enforced per game round as opposed to per game session.

The third player in KissKissBan is called the *Blocker* and is competing with the Couple players. Before each round, the Blocker has seven seconds to enter as many words as possible which the Couple players are not allowed to use. In contrast to the taboo words in the ESP Game, the Couple players cannot see this list of words. If any of the players enters a blocked word, five seconds are subtracted from their remaining time. If the timer runs out before the Couple players achieve a match, their scores are decreased and the Blocker's score is increased. If the Couple players succeed, the opposite is the case.

Results. The fact that KissKissBan has a competitive element makes the analysis of the gameplay more difficult: Since the Couple players do not know the blocked words, they must adjust their guesses according to their assumptions about the Blocker. In turn, the Blocker is trying to block terms which the players are likely to use. KissKissBan is therefore an

imperfect information game and the equilibrium[5] between the players will determine the created labels. Ho et al. do not theoretically analyze their game design. They do argue, however, that their approach will yield more diverse labels.

A certain advantage of KissKissBan's design is that cheating is even more difficult than in the ESP Game. The results can only be manipulated if all three players cooperate. If only the Couple players work together using an agreed-on strategy, the Blocker will understand that strategy and block it in future rounds. A Couple player and the Blocker cannot cooperate, since only terms which the second member of the Couple has also entered are accepted.

The invisible list of blocked terms is identical to the suggestion of Weber et al. [32] to hide the list of taboo words. In Section 1.3, it is argued that this approach will frustrate players quickly. This should be less of an issue in KissKissBan. Firstly, the punishment for using a blocked term is far less severe (subtraction of time as opposed to failure of the entire round). Furthermore, the Couple players will not be punished by an algorithm, but by a human player who gains in return. This creates a competitive experience that could not arise when playing against a computer generated list of words and that reduces the potential frustration.

2.2.5. Phetch

Von Ahn et al. argue that the labels created by the ESP Game are very well suited for image retrieval, but are insufficient as actual descriptions of an image. Such descriptions could for example be used to make the Web more accessible to the blind. To solve this problem, von Ahn et al. propose Phetch, an image labeling game targeted at collecting entire sentences describing an image [28].

Basic Gameplay. Phetch is designed to be played by three to five players. One of the players is randomly selected as the *Describer*, whereas the other players form the group of *Seekers*. A picture is shown to the Describer, who can enter arbitrary sentences to describe the image to the Seekers. The Seekers must then use a special search engine to locate the described image. To do this, they can enter arbitrary search queries. If a Seeker selects the correct image from her list of results, she and the Describer are awarded a score bonus. To discourage random guessing, points are deducted whenever a player ventures a wrong guess. After the correct image has been found, the Seeker who found it becomes the Describer in the next round. An overall time limit is enforced.

The time it took the Describers to find the correct image is used as in indicator of the quality of the given description. Like all other games by von Ahn et al., Phetch allows single players to participate using simulated opponents. Describers are simulated by replaying recorded descriptions, which also provides additional verification. This process can be repeated arbitrarily to increase the accuracy of the descriptions. Seekers are simulated using the keywords assigned to the images. The simulated Seekers only guess correctly if a minimum percentage of the labels assigned to an image were used in the description given

[5] see Section 2.4.1 on game theoretic analysis of GWAPs

by the human player.

Results. Because of the time constraints imposed on the players, the collected descriptions can contain errors and are often given in casual, chat-like language. Von Ahn et al. consider this acceptable, because even misspelled descriptions constitute an improvement over the previous state of the art.

The goal of PHETCH is to collect descriptions of images that are verbose enough to – for example – allow blind people to understand the contents of an image. To evaluate whether this goal was achieved, von Ahn et al. performed a user study. The participants of this study were divided into two groups. A grid of 30 images was shown to the members of each group, which were selected based on the similarity of their respective labels from the ESP GAME. The participants were then asked to select a specific image from the grid, based on a given description. For the first group, this description consisted of the labels assigned to the image by the ESP GAME. The members of the second group were given a description generated by PHETCH. The first group succeeded 73.5% of the time, whereas the group relying on PHETCH descriptions selected the right image in 98.5% of all tries. Von Ahn et al. conclude that the captions created by PHETCH are significantly more descriptive than those generated by the ESP GAME.

2.2.6. Additional Image Labeling Games

Apart from the games described above, a number of other implementations exist. Discussing all existing Games With A Purpose in detail exceeds the scope of this thesis. The following implementations are therefore described in a more concise form. While some of the games described below are not strictly image labeling games, they are listed here because of their close relation to image retrieval and image labeling.

- MATCHIN by Hacker and von Ahn [10] is a two-player output-agreement game. Two images are shown to each player. The pair of images is identical for both players. The players must each choose the image which they think their partner prefers. If they agree, the players are awarded a score bonus.

  Hacker and von Ahn use the collected *relative* preferences to calculate a *global* ranking of the images, according to perceived beauty. This ranking has applications in computer vision (for example, for automatically evaluating the beauty of images) as well as image retrieval (for retrieving appealing images first).

- PICTURE THIS by Bennett et al. [1] is a two-player output-agreement game, designed to improve the results of image search engines. A random search query and a number of resulting images retrieved by a search engine are shown to each player. The players' goal is to agree on which image is the best result for the given query. The order of the images is randomized to avoid biasing the players. In the first round of the game, only two resulting images are shown. Achieving a match is therefore relatively easy. After each successful round (meaning rounds in which the players achieved an agreement) the number of images in the next round is increased. Conversely, the number of images is decreased if the players fail to agree. Thus, the difficulty of the game is

automatically adjusted to match the skill of the players.

- Magic Bullet by Yan and Yu [33] is a four player game, designed to decipher CAPTCHAs. The players are randomly assigned into two teams of two players each. The players in a team cooperate with each other, but compete with the other team. A single letter extracted from a CAPTCHA is shown to all players. The goal of each team is to arrive at an agreement about which letter is depicted by pressing the corresponding key on their keyboard. The first team to arrive at such an agreement is awarded points. To increase the appeal of the game, the authors visualize the CAPTCHA character as a bullet, heading towards the goals of either team. Yan and Yu provide anecdotal evidence that the players enjoy the competitive gameplay.

2.3. Non-Image-Labeling Game Designs

While this thesis focuses on Games With A Purpose used for labeling images (see Section 2.2), there is a large number of different problems for which Games With A Purpose have been implemented. Below, a selection of these different game designs is described.

2.3.1. Verbosity

Aside from a number of different image labeling games, von Ahn et al. created Verbosity, which is designed to collect basic truths about the world [29]. Assertions like "A cat is an animal" are common-sense knowledge to almost every human, but are mostly inaccessible to computers. A large collection of such truths, combined with automatic reasoning could help computer systems to behave far more intelligently. However, many such truths are so trivial that they have never been recorded. Manual entry of such statements is tedious and error-prone. Collecting these data with a game is therefore a feasible alternative.

Basic Gameplay. Verbosity was inspired by the party game Taboo[6]. Thus, a first fundamental difference in gameplay between Verbosity and the ESP Game is that the individual players assume different roles in Verbosity. The game assigns one player to be the *Narrator* and one player to be the *Guesser*. A random, secret word is shown exclusively to the Narrator. The goal of the players is to make the Guesser correctly enter the secret word. To do this, the Narrator can use sentence templates to describe the secret word. Example templates include "The secret word is a kind of ..." or "The secret word is used for ...". The Narrator can enter one or two words replacing the dots. The use of templates allows automatic parsing of the collected truths and avoids the ambiguities of free natural language. Furthermore, the templates allow the classification of the information the player has entered. For example, truths like "A wrench is a kind of tool" and "A wrench is used for fastening bolts" relate very different information about a wrench.

Aside from these templates, the players do not have a communication channel. Therefore, the only way for the Guesser to correctly enter the secret word is to deduce it from the

[6] In Taboo, published by Hasbro, two teams of players compete against each other. In turns, a player from each team tries to describe a word given on a card to her teammates, without using any words from a list of taboos given on the card [Hasb].

truths entered by their partner. Thus, if a word has been entered correctly, it is reasonable to assume that the truths entered before are valid for the given object.

Replay Rounds. Like all other games developed by von Ahn et al., VERBOSITY allows single players to participate in the game. Simulating a Narrator is achieved by replaying truths that were collected during previous rounds. In contrast to the ESP GAME, in which entire rounds are replayed, VERBOSITY selects facts from different rounds. This ensures that the verified facts are independent of other truths mentioned in the game session. A score is assigned to each fact collected in the game, with an initial value of zero. When a human Guesser successfully guesses a secret word, the scores of the facts entered by the Narrator are increased. Once the score of a fact exceeds a certain threshold, the fact is considered valid.

Like PEEKABOOM (see Section 2.2.2), VERBOSITY faces the challenge of simulating a player that has to react to the human player's input (in this case, the Guesser). In VERBOSITY, this is achieved by compiling a list of related words for every secret word. Once a human Narrator has entered enough of these related words, the game "guesses" correctly.

2.3.2. Foldit

Cooper et al. introduce a new class of Games With A Purpose, which they call *scientific discovery game* [5]. In common GWAPs, the underlying problem is easy to solve for a human. For example, labeling an image is simple and the goal of an image labeling game is primarily to motivate players to perform the given task. In contrast, the underlying problem in scientific discovery games is difficult to solve, especially for laymen. The goal of a scientific discovery game is therefore to translate such a difficult scientific problem into a set of puzzles which can be solved by non-expert users. The solutions to these puzzles must then be usable to advance the solution of the underlying problem.

To demonstrate this approach, Cooper et al. have created FOLDIT. The functional goal of FOLDIT is to find the *native structure* of a given protein. Proteins are chains or *sequences* of amino acids that fulfill various functions in living organisms. These chains fold into their *native structures*, which are the spacial arrangements with the least free energy. To predict the function of newly created proteins, computationally predicting their native structure is vital. Calculating the free energy of a given configuration is relatively simple, but because of the very large number of degrees of freedom minimizing this energy computationally is very difficult. Cooper et al. strive to solve this using a Game With A Purpose.

Basic Gameplay. In FOLDIT, players can choose from a set of puzzles and try to solve any one of them without time constraints. Each puzzle consists of a single protein for which the native structure should be found. This protein is displayed in a relatively abstract fashion as a three-dimensional object. Users can arbitrarily rotate the object to view occluded parts. The free energy of the protein is translated into the player's score – the less energy, the higher the score. To increase their score, players can manipulation the structure of the protein. The primary mode of interaction is clicking and dragging using the mouse, allowing

the user to directly manipulate parts of the protein. Furthermore, automatic tools can be invoked by the player to perform global optimizations on the protein automatically.

Games like the ESP GAME only need minimal introduction. In contrast, FOLDIT's user interface can be overwhelming to new users. Therefore, Cooper et al. have created a set of introductory puzzles that incrementally show players how to use the tools provided by the game.

Results. In all other Games With A Purpose previously described in this Section, the results of the individual players can automatically be combined into data which is correct with arbitrary likelihood. This is not the case in FOLDIT. Because of the large number of degrees of freedom in a protein, no two individual solutions can be expected to be identical. Thus, no majority voting can be applied to combine the different results. Furthermore, since the users are usually not experts in protein folding, the only indication of correctness for a given result is the score that is automatically calculated. Because of the complexity of protein folding, some properties of a proteins structure are not captured by this score. Because of these issues, the results generated by FOLDIT players are clustered and ranked and presented to experts for further inspection. The results of this selection process were compared to the actual native structures of the respective proteins. In a contest with other prediction methods, Foldit delivered mixed results. In some cases, Foldit performed better than other methods, in others it performed worse. Nevertheless, Cooper et al. conclude that they succeeded in creating a game that helps to solve protein folding problems.

2.3.3. TagATune

TAGATUNE is a GWAP for tagging audio files, created by Law and von Ahn [14]. Law and von Ahn experimented with using the game mechanics of the ESP GAME to create tags for audio files. A prototype revealed several problems with this approach. It is apparently far more difficult for two players to agree on a label for an audio clip than for a picture. As a result, in 36% of the rounds played in the prototype the players opted to pass. Furthermore, the enjoyability rating of the prototype in a user study was only 3.4 out of 5. To solve this issue, Law and von Ahn developed a novel game mechanic called *Input-agreement* (see Section 2.1).

Basic Gameplay. Like most Games With A Purpose, TAGATUNE is a collaborative, two-player game. In each round, a short clip of music is played to each player. This clip can be – but is not always – identical for both players. The players can describe their clip using arbitrary text. These descriptions are visible to both player. Each player can then select if she believes that her clip is identical to that of her partner or she believes it is different. If the players agree and they have chosen correctly, they are awarded a score bonus. Since both players have to make the right choice to be rewarded, both players must strive to create accurate descriptions of their piece of music. If the players agree, the descriptions of both players can be considered correct for the respective songs. Even so, players only have to agree on a binary choice, which is easier to achieve than an agreement on labels, leading to a more enjoyable game.

Results. In contrast to the 36% of all rounds that were passed on in the prototype (using *Output-agreement*), only 0.5% of the rounds in the final version of the game (using Input-agreement) were passed on. Furthermore, 80% of all played rounds were successful, meaning the players agreed on the correct answer.

Another interesting property of Input-agreement verification is its lack of cheating strategies. No player holds the ground truth of the game (the knowledge, whether or not the two songs are identical). Thus, free textual communication between the players is allowed and there is no communication barrier that could be circumvented. Furthermore, players cannot agree on a strategy before playing the game, since any strategy that does not consider the input of the game (the played music), can only lead to a 50% success rate (because players not only have to agree, but agree on the correct answer).

2.3.4. People Watcher

People Watcher, introduced by Paek et al., is a GWAP aimed at creating alternative phrasings of business names for use in automated directory services [20]. The interesting difference to other GWAPs is the way this intention is communicated to the players. For example, the ESP Game does not explicitly prompt the players to describe the displayed images, but it does not hide its intention of collecting labels either. In contrast, People Watcher claims to be designed to collect very different marketing data.

Basic Gameplay. People Watcher is a two-player collaborative game. A pair of telephone directory business listings is shown to each player. The players have an unlimited amount of time to memorize the listings. Once they have done so, they can choose to proceed. Two pictures depicting one or more persons are shown to the players. The players' task is to decide which persons are likely to be customers of a given business. The photograph of the respective persons must be assigned to the name of a business shown before. This has to be accomplished by typing the name of one business underneath each photograph. After completing this step, the players are shown their partner's choice and are rewarded points for matching answers. An overall time limit is enforced.

Results. The gameplay described above is an example for an Output-agreement game. However, the important difference lies in the data that is collected. Instead of collecting the association of photographs and business listings, People Watcher forces its players to memorize and repeat the given listings. Paek et al. argue that players reproducing a listing from memory generate a gist of the original text. Furthermore, the time limit encourages players to give brief answers. As the players have to agree on their descriptions, they are motivated to provide reasonable answers.

An evaluation of People Watcher performed by Paek et al. showed that the game fulfills its goal of creating reasonable alternative phrasings of directory listings. However, the participants rated the fun of playing the game at only 2.5 on a 5 points Likert scale.

2.3.5. Additional Non-Image-Labeling Games

As was the case with image labeling games, too many implementations of non-image-labeling GWAPs exist to be discussed in detail in this thesis. Nevertheless, the following game designs contain interesting solutions. They are therefore described concisely instead of dropping their description entirely.

- OntoGame by Siorpaes and Hepp [21] is a series of games designed to collect data for *Semantic Web* applications. All games are output-agreement games for two players. Possible player tasks include – among others – annotating videos from Youtube [you] and aligning entities gathered from Wikipedia [wik] to so-called ontologies, thus creating machine-readable descriptions of real-world concepts.
- CityExplorer by Matyas et al. [19] is designed to collect images, geographic positions and descriptions of points of interest in real cities. Each round of the game can be played by an arbitrary number of players and can last from a few hours up to several weeks. Before a game round, each participating player can describe several location categories – for example "church" or "bar". To win the game, the players need to gain points by claiming points of interest. To do this, players have to possess a smartphone capable of running the game client, taking pictures and determining its position through GPS[7]. Using this smartphone, players need to take a picture of a point of interest that falls into a category named before the game. The client software automatically determines the player's position and sends it to the central game server along with the photo. To verify this data, a second layer of the game runs concurrently. Players who are not participating in the first layer can review the submitted photos and locations. If a location is accepted by an independent player, the player who originally submitted the point of interest is awarded a score bonus. Using this mechanism, CityExplorer collects verified locations and photos of various points of interest.
- Page Hunt by Ma et al. [17] is different from the previously described games, in that it is designed to be played by only a single player. Its goal is to improve the results of Web search engines. In each round of the game, the player is presented with a randomly selected Web-site. Her goal is to formulate a query to a search engine, which will lead to the Web-site being returned in the first N results. If this is the case, a new site is selected and the player is awarded a score bonus. The player can enter an arbitrary number of queries, but an overall time limit is enforced. Using this approach, the game collects search queries that players deem appropriate for a given Web-site. These can be used to improve search engines to better fulfill the expectations of their users.
- Intentions by Law et al. [15] is also designed to improve search engines. To return the most appropriate results for a query, a search engine should infer the intention that lead to it. As an example, Law et al. describe the query of "Greyhound Bus", the intention of which could be to find the current schedule, the stock price of the company or the nearest bus stop. The goal of Intentions is to gather training data to

[7] The Global Positioning System makes it possible to determine ones current location anywhere in the world with an accuracy of a few meters.

automatically infer the most likely intention from a given query. The authors propose to do this using a two player, Input-agreement game design. An intention in form of a question such as "How do you treat Parkinson's disease?" is shown to each player. The players can each enter a search query, the results of which are shown to both players. The players must then agree on whether or not they were given the same intention. This approach forces players to translate intentions into related search queries. Law et al. propose to use the collected data to create linguistic patterns that describe how users translate intentions into queries.

2.4. Related Meta-Articles

While most literature on Games With A Purpose focus on the design and implementation of novel games, some articles review existing implementations or propose general models. The following Section describes two papers falling into this category of meta-articles.

2.4.1. Game-Theoretic Analysis of the ESP Game

Jain and Parkes present a game-theoretic model of the ESP Game [13]. The goal of this model is to formalize and prove which playing strategies are most successful in the ESP Game and what the implications for the resulting tags are.

Jain and Parkes analyze the most basic version of the ESP Game, in which the scores players receive for a match are independent of the matched word. Therefore, the goal of the players is to complete as many rounds as possible within the given time limit. This in turn implicates that players strive to match as early as possible in each round. Jain and Parkes call this *match-early preferences*. They furthermore assume no taboo words are used in the game.

The model designed by Jain and Parkes assumes that a describing set of words exists for each image (which the creator of the game is trying to learn). The words in this set posses and are ordered by a *frequency*, which defines the likelihood of a word being assigned to the given image. Before the actual game, the players are modeled to each choose an *effort level*. This effort level defines the subset of words, from which a player samples when describing an image. For example, a *low effort* player will only use the n most frequent terms to describe an image, whereas as a *high effort* player will select from all available words.

Jain and Parkes then proceed to prove that choosing *low effort* and playing the describing terms in order of descending frequency leads to a Bayesian-Nash equilibrium for the ESP Game. In such an equilibrium, no player can gain an advantage by changing their strategy. Thus, the authors have proven that the basic version of the ESP Game tends to lead to common tags. The analysis of taboo words and incentive structures which would lead to uncommon tags is left as future work.

2.4.2. Secure Distributed Human Computation

Gentry et al. propose a framework for what they call *secure distributed human computation* (SDHC). An SDHC system is constituted by four parties:

- A set of *problem suppliers*, who have a set of problems they need solved. This is usually a company.
- A set of (human) *problem solvers*, who are capable of solving such problems. These are the primary users of the system – the players, in case of a game.
- A *distributor* who assigns problems to solvers who are willing to process them.
- A set of *store fronts*, which are venues (usually Web-sites), at which solvers can contact the distributor to acquire tasks. The store fronts usually provide some kind of service to the solvers as a reward for processed problems.

Gentry et al. divide the interaction between these entities into a registration and an operation phase. The authors furthermore propose a threat model for both phases. In the registration phase, the primary attack lies in the (possibly automated) creation of multiple user accounts. In the operation phase, the primary attack lies in the introduction of false answers into the system by malicious users.

From this, the authors derive a probabilistic analysis of the reliability of a SDHC system. To achieve any reliability in human computation, each problem should be processed by multiple users. Under the assumption that there is only one correct answer for each problem, the answers of the users must be combined to reach an overall conclusion. Gentry et al. compare majority voting and Bayesian inference for achieving this. In a majority vote, the answer given by the highest number of users is elected to be the correct one. In contrast, Bayesian inference takes into account the previous results of a user, thus putting more trust into answers by users that have given right answers before.

Gentry et al. prove that majority voting yields the most reliable results in the presence of malicious users. Assuming that malicious users only provide false answers and the other users are correct 75% of the time, only two thirds of all players need to be honest for majority voting to succeed. If these conditions are met or exceeded, the reliability of a solution increases exponentially with number of users assigned to a given problem.

3. Design of the Proposed KARIDO Game

As explained in Section 1.3, the primary motivation for the design of the new image labeling game called KARIDO is increased tag diversity. This Section of the thesis describes the core mechanics of the proposed game and gives arguments for their influence on the diversity of the collected tags. Furthermore, the reasoning behind a number of key decisions in the design of the game is explained. The last part of this Section contains an overview of the disadvantages of the new design and argues why these disadvantages are acceptable trade-offs.

3.1. Overview of KARIDO Gameplay

In its basic characteristics, KARIDO is similar to VERBOSITY (see Section 2.3.1). Both games are inversion-problem games, in which two players assume the roles of *Describer* (called *Narrator* by von Ahn et al.) and *Guesser*.

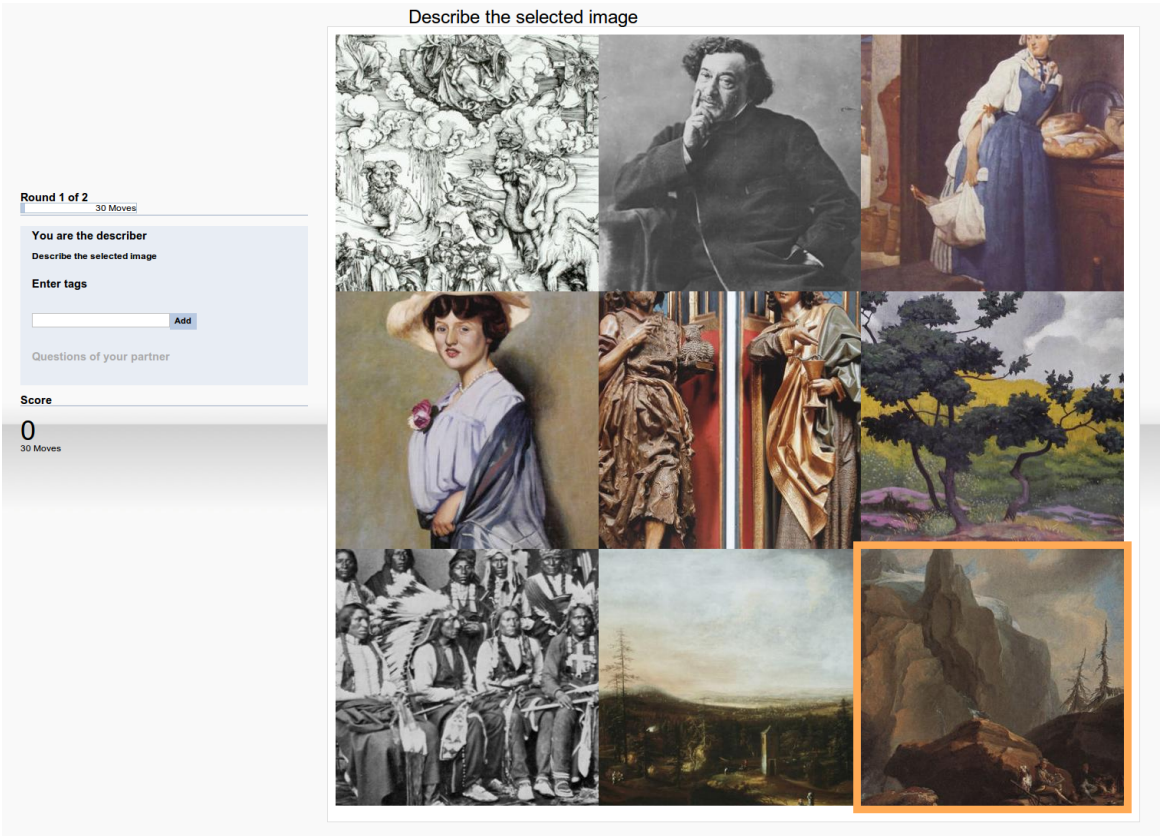


Figure 4: Screenshot of KARIDO from the view of the Describer

Before each round of KARIDO, a set of N similar images ($N = 9$ in the current implementation) are randomly selected. The selection of these images is crucial to the goal of increasing tag diversity and is discussed in more detail in Section 3.1.1. These images are displayed as regular grids to both players (see Figures 4 and 5). In the view of the Describer,

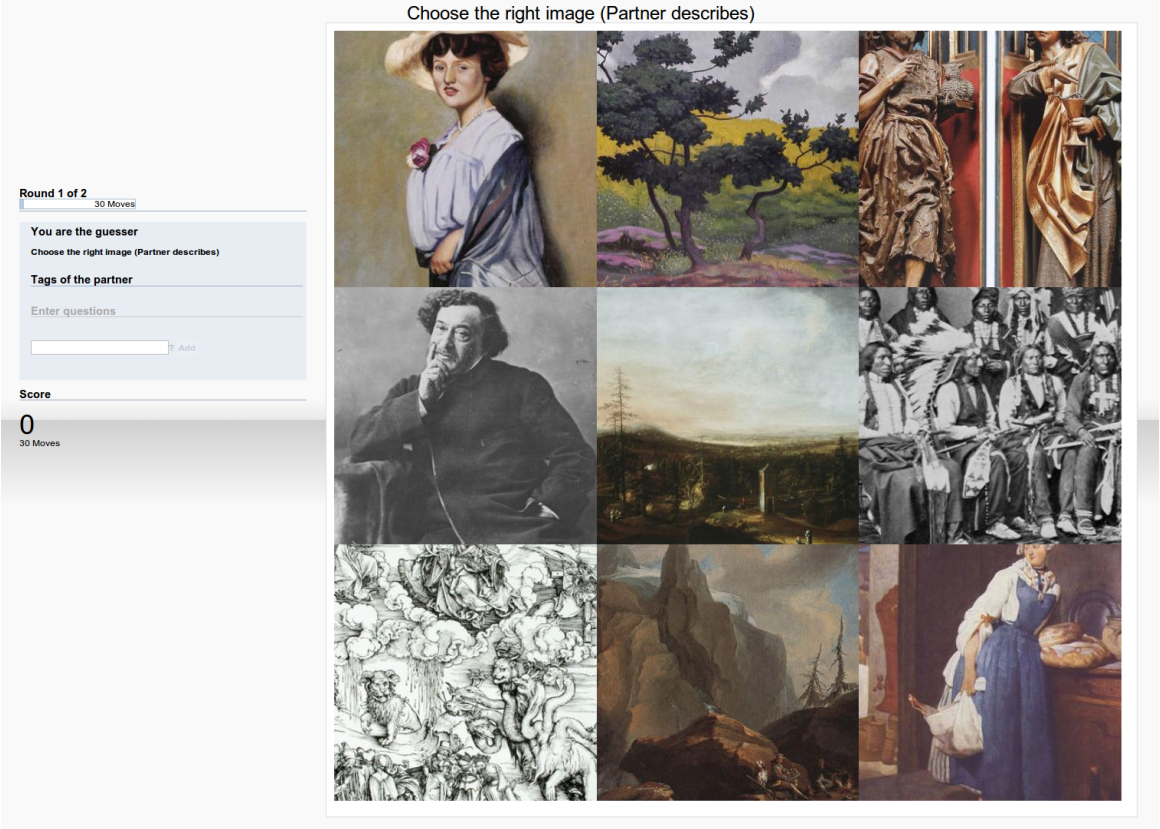


Figure 5: Screenshot of KARIDO from the view of the Guesser

a single image is highlighted (this is called the *goal*). The Describer's task is to describe the goal image to the Guesser. To achieve this, the players need to communicate in some way. As KARIDO is an inversion-problem game, there are few restrictions that must be enforced on player communication. There are therefore several possible forms for the communication channel between the players, which are discussed in Section 3.2.2.

The Guesser's view shows the same images as the Describer's, but in randomized order and without a highlighted image. The Guesser's task is to deduce the current goal image from the given description. At any time during the game, the Guesser can click any image twice to guess it. If this guess is correct, the image is removed from the grid of both players and both players receive a score bonus. Furthermore, the tags entered by the Describer are considered valid (see Section 3.2.1). The Describer must then select another image, which becomes the next goal. Such a cycle of the game (from selection of a new resource to selection of the correct goal image) will from now on be called a *turn* of the game. The game proceeds in this fashion until only one image remains. Since selecting this image would be trivial, the current *game round* is ended and a new round is initiated (with a newly selected set of images).

In other GWAPs such as the ESP GAME or VERBOSITY, the players enter their answers in form of textual labels. Therefore, the set of possible answers contains all character sequences and thus millions of items. To produce a match, a player must enter one of only

a few items from this set: In case of the ESP GAME, the winning labels are those tags that the player's partner has already entered, in case of VERBOSITY, the only winning label is the input given to the Narrator. The important point is that it is not possible for a player to try random answers, because the probability that such an answer is correct is very small.

In contrast, the Guesser in KARIDO can only select from a given set of images. The number of images in this set must be relatively low for several reasons. For one thing, the resolution and size of current computer displays limits the number of images that can be displayed at the same time. Furthermore, since the Guesser must select a specific image from the set, she should have a mental overview of all images. This also holds true for the Describer, who must differentiate between all images in the set. In an effort not to overwhelm the players, only nine images are displayed. Using larger sets is possible, but it seams doubtful that players would enjoy having to consider 20 or more images at the same time.

Consequently, the number of potential answers of the Guesser is below 20. It would therefore be possible for a player to try all possible answers. This would eliminate the verification of the entered tags, since a player relying solely on guessing can ignore the description given by her partner. It is therefore necessary to take measures that discourage random guessing. In KARIDO, the score of both players is reduced as a penalty for selecting a wrong image. A detailed description of this mechanism and the possible alternatives can be found in Section 3.2.4.

Finally, to make the scores of players comparable and to create a goal for the game, a limit on the duration of each *game session* must be enforced. A session is the time in which two players remain in the game after being matched. A game session usually consists of several rounds. If the length of a game session were not limited, the player with the most endurance would gain the highest score. Furthermore, players would have to play until they have actually had enough and quit the game of their own accord. This would most likely leave the players with a negative impression. By limiting the duration of each game session, the goal of the game can be defined as "score as many points as possible in the given time". This clearly defined goal should incite players to put effort into the game, resulting in turn in more appropriate output labels. Malone states that a goal is one of the most important properties of a fun game [18]. A duration limit can be enforced in several ways, which are discussed in Section 3.2.3.

3.1.1. Input-Similarity

The primary goal of KARIDO is to collect more diverse tags than previous image labeling games. The key element for achieving this goal is what will from now on be called "input-similarity". Despite the fact that it uses inversion-problem verification, KARIDO is similar to input-agreement games such as TAGATUNE (see Section 2.3.3). In input agreement games, the difficulty of the game largely depends on the similarity of the input data given to the players, as described by Law and von Ahn [14]. Agreeing on whether two input resources are identical is easy if the resources are either identical or complete different. In contrast, describing assets that only differ in nuances is far more difficult.

The same principle applies to KARIDO. If the images in the grid are highly diverse, it is

easy for the players to describe and select the goal image. For example, consider a grid in which each image contains only a unique single color. In this case, the Describer can use the color to clearly describe the goal. In contrast, the more similar the images are, the more difficult describing and guessing correctly becomes. Given a grid in which all images contain a red car, the Describer has to find characteristic traits of the goal image that differentiate this image from all others. Thus, the difficulty of the game can be adjusted by selecting similar or diverse images.

However, adjusting difficulty is only a secondary effect of input-similarity. By selecting similar images in KARIDO, the players are forced to find differentiating properties of the images. Thus, without the use of explicitly restricting methods such as taboo words, players must contribute new information. Law and von Ahn describe a similar behavior for input-agreement games [14].

What remains is to describe how the intuitive notion of "similarity" can be formalized and how a computer can automatically select similar images. Using image analysis is an interesting option, but is currently unlikely to provide fully satisfying results. For example, distinguishing a red car and a red truck is easy for most humans, but very difficult for computer vision systems. Furthermore, relying only on image similarity will yield static results (as the images don't change). Thus, the similarity of the images in the sessions of the game would have to be adjusted dynamically to yield both general and specific tags.

KARIDO relies on the simpler notion of using player-created tags as a measure of image similarity. For example, two images tagged as "red", "car", "Ford" and "red", "car", "Mercedes" are considered more similar than a third image tagged "red", "dress" (the first two images share two tags, whereas the third image and the first two only share one). This means that for a new set of images without any tags, all images are considered equally similar. Therefore, the images in the grid are selected randomly and can be expected to be relatively diverse. As a result, players can use more general terms to distinguish the images and will therefore create such tags for the images. This means that some images are now more similar than others. For example, the photographies in a collection could be tagged "photo", whereas paintings could be tagged "painting". Thus, the players will be given grids which contain either only photographies or paintings. Therefore, the attributes "photo" and "painting" can no longer distinguish the images and the Describers are forced to find new properties of the images.

If any two images exist that do not have a distinguishing tag, they will necessarily both be selected for a game round at some point. One of the images will necessarily become the goal image, while the second image is still in the grid. Thus, the players either have to come up with a label that distinguishes these two images or use random guessing. As the scoring of the game is designed to make random guessing an infeasible strategy (see Section 3.2.4), a player who is able to find a distinguishing feature is likely to use this trait to describe the image. Thus, the process of refinement of the labels continues until all images possess a set of tags that sets them apart from all other images in the game. Potentially, there could be images that are not identical but are indistinguishable for a human – for example, two highly similar digitized versions of a painting. However, distinguishing such images is beyond the scope of KARIDO.

Following the reasoning described above, the tag sets collected by KARIDO will converge

towards a state in which they contain both general tags as well as specific ones.

3.2. Important Design Decisions

As describe above, the primary expected advantage of KARIDO's design is the automatic emergence of comprehensive tag sets for all images of a collection. However, a secondary advantage lies in the flexibility of the design. Some elements of existing Games With A Purpose are dictated by the necessities of the game mechanics and cannot be changed without severely altering the game. For example, output-agreement games cannot allow communication between the players.

Most of the gameplay elements in KARIDO can be changed easily without altering the overall game design. This flexibility makes it possible to build several versions of the same game and compare their performance in terms of the resulting tags and the satisfaction of the players. It should be possible to deduce which gameplay elements lead to the best performance from the results of this evaluation. The following Section describes elements of the game which allow several design choices. Furthermore, the options chosen for the implementation of the game are described, along with the reasoning behind these choices.

3.2.1. Tag Verification Mechanisms

Like most GWAPs, KARIDO considers data entered by one player unreliable until it is independently verified by at least one other player. As KARIDO is an inversion-problem game, the output of the Describer is considered valid once the Guesser has selected the right image. The correct selection implies that the set of labels entered by the Describer is relevant in some way to the goal image.

The simplest approach to model the relevance of a label is to associate it with the goal image and count the number of associations for any labels that have been used several times. The number of associations between a tag and an image can be used as a measure of the relevance of the label. This approach has been used successfully by von Ahn and Dabbish in the ESP GAME [25].

However, KARIDO collects further data during the game. Some of this data can be used to judge the relevance of a label for an image more precisely. In contrast to the ESP GAME, KARIDO can collect several labels for an image in a turn of the game. The Describer usually enters several tags before the Guesser correctly selects an image. Thus, the labels first entered were not sufficiently precise to enable the Guesser to select the correct image. It is therefore reasonable to assume that the labels assigned later are more relevant to the image. KARIDO uses a weighting scheme to take this increasing relevance into account. All tags have a base relevance weight of 0.5. An additional weight of 0.5 is distributed over all tags in a linearly increasing fashion (see Figure 6). Thus, more weight is given to tags which are assigned later. At the same time, the base weight ensures that all assigned tags are considered at least slightly relevant. The plot of the weighting algorithm also shows that the increased weight of later tags diminishes as more tags are added.

As explained in Section 3.1, the probability that two players of the ESP GAME create matching labels using random guessing is very low. In contrast, the probability of a Guesser

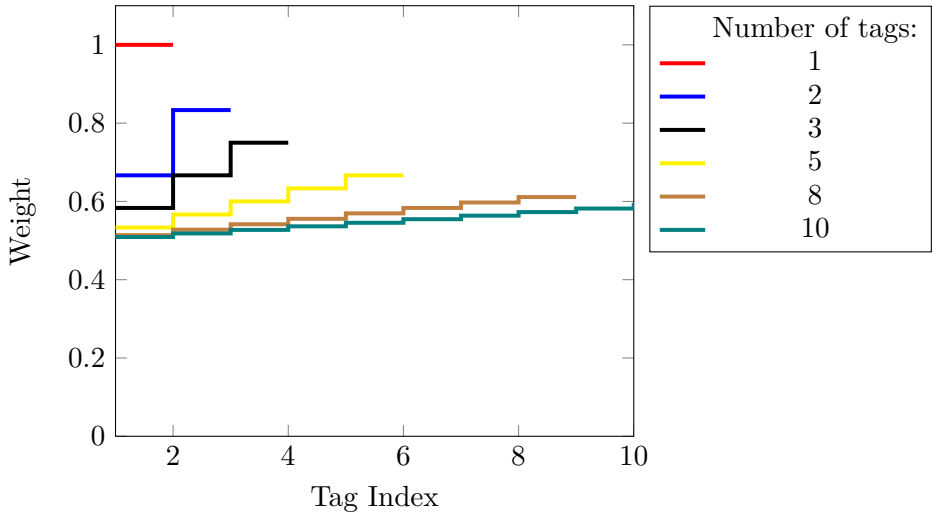


Figure 6: Relevance weights assigned to tags as a function of tag index (the first tag entered has index 1) and total number of tag entered (at the time the goal image is selected)

in Karido selecting the right image by chance is relatively high. The scoring system ensures that a player using random guessing cannot expect positive scores. However, players can still use random guessing and will eventually select the right image. In this case, the relevance of the assigned labels should be considered zero. If the labels were considered valid, players could introduce wrong data into the system.

To avoid gathering wrong data, the relevance weight of tags in Karido depends on the number of wrong guesses performed before the selection of the goal. If the Guesser has tried more than 30% of the images before selecting the right one, all tags are considered irrelevant. For example, at the beginning of a game (with 9 images remaining), the Guesser must be right on the third try or the labels will be discarded. However, the full weight is only assigned if the guesser is right on the first try. Otherwise, the weight is interpolated linearly between 1 and 0 depending on the percentage of wrongly selected images (from 0% to 30%).

Another possibility for calculating the relevance of tags in the face of wrong guesses exists. It would be possible to assign negative weights to tags if they lead to a Guesser selecting the wrong image. This would reduce the relevance rating of tags with little descriptive value. However, this scheme could be exploited to introduce wrong data into the system: It is trivial to guess the wrong image on purpose. Thus, even a single malicious Guesser could decrease the scores of valid labels by guessing wrongly. Therefore, no negative weights are ever assigned to a tag.

The two measures described above are combined by calculating the product of the two individual weights. Thus, the relevance weight assigned to a tag in a turn of the game is between 0 (if the Guesser selects more than 30% of the images wrongly) and 1 (if only one tag is assigned and the Guesser immediately selects the right image). This more nuanced measure of relevance cannot be modeled using only integer values (such as the number of

times a tag as been applied). To avoid this problem, KARIDO uses an approach similar to the system created by von Ahn et al. for VERBOSITY [29]. The first time a label is assigned to an image, a score for the relevance of this label for the image is created and initialized to zero. This score is a real-valued number and is increased by the calculated weight each time a tag is verified. As described above, the calculated weights are always greater or equal zero. Thus, the relevance score of a tag is increasing monotonously. This implies that if any tag erroneously gains a high relevance score, this error will never be revised. As described above, decreasing the scores of tags is not a viable alternative. Additionally, the verification scheme of the game ensures that the probability for increasing the relevance scores of irrelevant tags is very low. Thus, by considering the scores as a relative measure between tags, relevant labels can be selected reliably.

The resulting relevance score could be used to order the labels by relevance or return the more relevant results of a search query first. Alternatively, a threshold value could be imposed and only labels with scores exceeding this value would be considered valid labels of an image. This threshold can also be defined relative to the entire collection, for example by defining a quantile of accepted labels.

3.2.2. Player Communication

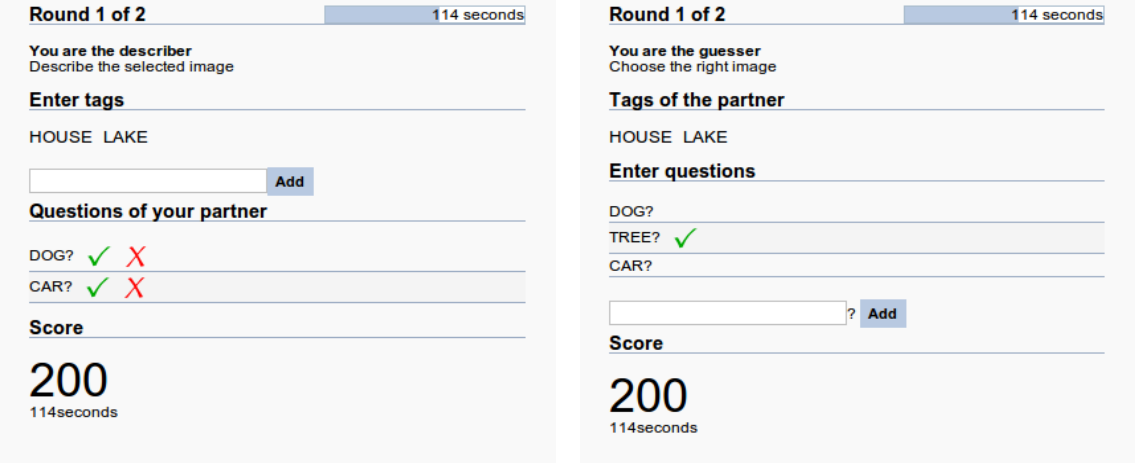
Most GWAPs rely on some sort of player agreement to verify that the entered data is correct. It is therefore essential that players do not have the possibility of artificially creating an agreement. The most common way of creating such an agreement is by using a communication channel outside of the game. Whether the communication between the players inside the game has to be restricted depends on the used verification method. For example, if even one of the players of the ESP GAME could send messages to her partner, the players could create arbitrary agreements. In contrast, games using inversion-problem verification can allow their players to communicate much more freely.

In KARIDO, the Describer has to communicate which image is currently highlighted. A unidirectional communication channel from the Describer to the Guesser is therefore the bare minimum requirement for a playable game. If no such channel existed, the Guesser would have to resort to random guessing, making the game pointless for both the players and the creator intending to collect data.

As the order of the images in the grid is randomized, it is not possible to describe the position of the highlighted image. Thus, the Describer has to communicate the contents of the highlighted image – which is exactly the data the game is trying to gather. Therefore, free textual communication from the Describer to the Guesser can be allowed. Additionally, the Guesser cannot influence the agreement other than guessing correctly and also holds less information than the Describer. thus, free textual communication from the Guesser to the Describer can be allowed as well. However, despite the increased reliability of inversion-problem verification, there are several ways to circumvent the verification scheme. These are discussed in Section 3.3.1.

Although completely free textual communication between the players could be allowed, it is reasonable to restrict the communication channel for several reasons. KARIDO is designed to collect image descriptions which could for example be used for image retrieval. This

means that the descriptions should be broken up into keywords for each image. Extracting appropriate keywords from free-form textual descriptions is a difficult problem of its own. One could argue that collecting more verbose descriptions is a valid goal (see also PHETCH, discussed in Section 2.2.5). In this case, accepting free-form input from the Describer can also be appropriate. For KARIDO, the text entered by the Describer is restricted to a maximum of three words and all punctuation is removed.



(a) Describer's view (b) Guesser's view

Figure 7: Player communication in KARIDO (screenshot)
Two tags and three questions have been created, one question has been answered

As discussed before, KARIDO can also allow a communication channel from the Guesser to the Describer. This channel can be used by the Guesser to help the Describer give more accurate descriptions. Usually, the Guesser will pose questions to reduce the set of possible images (for example "Is the image a photography?", "What kind of tree is there in the image?"). This introduces a new issue when collecting the descriptions: The Describer's text will now contain answers to her partner's question. In some cases, these can be legitimate descriptions of the image, that can be used independently of the question (for example, "The tree is a birch"). In contrast, many answers will be "Yes" or "No", adding no relevant description to the image if considered without the respective question. To integrate the questions and their answers into the collected labels and still allow the Guesser to take part in the description process, KARIDO allows posing Yes/No-questions, which the Describer can answer by clicking on an icon to send the appropriate answer instead of typing (see Figure 7a). While the Guesser sees all questions posed in the current turn (along with the answers of the questions that have already been processed, see Figure 7b), the Describer sees only the questions that she has not yet answered to avoid distraction.

In addition to aiding the Describer, these Yes/No-questions enable the Guesser to actively take part in the labeling process: Whenever a question is answered with "Yes", the question is added as a label to the description of the current goal image. If a question is answered with "No", a negative label could be attached to the current image. However, because of the limited informative value of negative labels (the list of things *not* depicted in any given

image is nearly endless), no tags are added for questions answered with "No".

3.2.3. Limits on Game Session Length

As described in Section 3.1, the duration of a game session must be limited to make the scores given to players comparable and to provide a goal for the players to achieve. There are several possibilities for limiting the duration of game sessions and game rounds. These approaches can be considered on two main "axes":

- The duration limit can be enforced for each round individually or for the entire game session.
- "Duration" can mean both the amount of time it takes for a game session to be completed as well as alternative measures, such as the number of actions taken by the players.

Approaches on these two "axes" can be combined arbitrarily, resulting in several potential limiting schemes.

Session and Round Limit. Limiting the duration of an entire game session is the most commonly used approach for GWAPs. For example, von Ahn and Dabbish propose a limit of 2.5 minutes for game sessions in the ESP Game [25]. In turn, neither the duration of a game round nor the number of game rounds in a game session is limited or fixed. This means that extremely slow players could play as little as a single round in a game session, whereas fast players can play a very large number of rounds in a single game. Thus, each game session has the same duration and a variable number of game rounds.

It is also possible to use the inverse approach of limiting the number of game rounds in a game session. This implies that a time limit has to be enforced on the individual game rounds or the length of a game session will be potentially unlimited. The result of using this approach is that the number of game rounds in each game session is fixed. The maximum temporal duration of a game session is defined as the product of the number of game rounds in a session and the maximum length of a single round. However, quick players can finish a game session in arbitrarily little time. In contrast to the aforementioned scheme, this approach does not yield a score bonus for quick players. As long as a round is finished within the given time limit, fast and slow players both play the same number of rounds and thus will earn the same number of points (assuming identically valued results in the game rounds).

As stated by von Ahn and Dabbish, enforcing a time limit is a viable tool for creating player motivation [26]. It seems likely that putting exceeding pressure on the players of a game might decrease the diversity of the tags used to describe the given images (as players who feel rushed will not take the time to carefully consider their input). The second approach of limiting the number of game rounds and the length of each round is therefore used in Karido. Each game session consists of two rounds. While this seems very little in comparison to the ESP Game, each round in Karido contains nine images and generally lasts longer than a round in other games.

Time and Action Metrics. As described above, time is not the only metric for limiting the duration of a game. However, because of the ubiquitous nature of time limits in Games With A Purpose and for comparison, a version of KARIDO in which each round is limited to 90 seconds has been implemented. As described above, game sessions in this mode contain two game rounds.

While time is a straight-forward metric for "duration", other metrics are viable alternatives. For example, in KARIDO, players communicate to describe images and try to guess specific images from a grid. These operations are the atomic *actions* of the game. Similar actions can also be found in most other GWAPs. Instead of limiting the temporal duration of a game session or round, it is also possible to limit the number of actions that players can perform.

For example, the ESP GAME could be changed to allow players to enter 30 tags in each game session, without limiting the available time. This would change the focus of the game. When a time limit is used, the winning strategy is to enter as many reasonable tags as possible in the shortest amount of time. This means that factors like typing speed or Internet latency can influence the performance of a player in the game. In contrast, by limiting the number of actions available to the players, the players can take as much time as they want for considering, typing and submitting their labels. This means that success in the game now only depends on whether the players can think of the same labels with as few tries as possible. As acquiring these matching labels is the purpose of the game, the number of actions (representing the number of labels that did not produce a match) is arguably a reasonable alternative to the metric of time.

However, using actions to limit the duration of a game poses a problem. Whereas time is always shared equally between the players of a game, more actions are "consumed" by players who perform more actions. Thus, a quick player can acquire a disproportionate share of the available actions. This introduces a potential competition between the players, which goes against the cooperative nature of most GWAPs. An equal distribution of the actions between the players should therefore be enforced. A simple option for achieving such equality between the players is to use a turn-based design. Most table-top games such as chess are turn-based, meaning that players make their moves by turns. For symmetric GWAPs, such as the ESP GAME, this limitation seems rather unreasonable. As the players cannot communicate, each player will, on average, spend half of the time in the game waiting for her partner to perform an action. Once this action has taken place, no visible change happens to the state of the game (aside from a notification that it is now the player's turn). The wait introduced by the turn-based mode will therefore likely seem arbitrary.

KARIDO is an asymmetric game, implying that the players can communicate relatively freely. Furthermore, the actions of the players directly depend on the actions of their partners. The player who is describing an image only has to clarify her description if her partner fails to guess correctly and the player who is guessing can only perform reasonable guesses in response to the given description. A turn-based mode with a limited number of actions is therefore well suited for asymmetric GWAPs and has been implemented in KARIDO. As in the mode using a time limit, the overall game duration is limited to two rounds per game session.

There are scenarios in which a rigid succession of player turns can be a problem. For

example, the Guesser in KARIDO has to select an image based on her partner's description. Occasionally, only two images are potential candidates. Thus, if the player takes her turn and guesses wrong, the correct solution is immediately obvious. Using a strict turn-based approach, the players would have to waste an action (the Describer's turn, which has become superfluous) and the Guesser would have to wait her turn until she could perform the now trivial action of selecting the right image. To avoid such situations, the Guesser in KARIDO can take a guess at any point in the game. While this approach allows the guesser to take a disproportionate share of the available actions, this is arguably an acceptable trade-off for the improved flow of the game.

Since the actual temporal duration of a game round is not limited in the mode using action limits, game rounds can potentially take a very long time to be completed. However, as the complexity of the game is relatively low (in comparison to chess, for example), most players are likely to perform their actions in a few seconds. Furthermore, there is no disadvantage if both players of a game session decide to take their time and play longer turns. Issues can arise if one player is considerably faster than her partner and has to wait what she perceives as a long time. In this case, the faster player could be frustrated and leave the game. However, the same problem exists when using a time limit and could only be resolved by trying to match players with a similar playing behavior.

3.2.4. Scoring and Penalties

As explained in the overview of KARIDO, it is necessary to discourage players from random guessing. It is furthermore necessary to reward and penalize both players equally. If only the Guesser were punished or rewarded, there would be no incentive for the Describer to put effort into her description. While it is imperative that random guessing must not be a viable strategy for playing the game (to ensure the validity of the collected tags), it is also important not to alienate players. Even if players act with the best intentions, it is highly likely that the Guesser will occasionally be wrong. If players are punished too severely for an event that happens regularly in normal gameplay, they are likely to be frustrated and to quit playing the game. As retaining players is highly important for any GWAP, frustrating players must be avoided.

The simplest and strictest possibility of penalizing wrong guesses is to end the current round for both players. This sharply disrupts the progress of the current game and makes it very unlikely that the players will ever complete a round of the game. Since finishing a game round will likely create a sense of achievement for the players, ending the current round as a penalty does not seem advisable.

Another potential penalty for random guessing is removing the current goal image from the grids of both players without awarding any points. This makes it very likely that the players are able to finish a game round, because this approach limits the number of turns in each round to $N-1$. This artificial shortening of game rounds makes it potentially too easy to finish a round and diminishes the sense of achievement of the players. This is supported by Locke et al. who state that task performance increases with task difficulty, as long as the abilities of the person processing the task are not exceeded [16].

Finally, it is possible to react to wrong guesses by only reducing the score of the players.

This leads to the concept of the *expected value* E of the players' scores for random guessing. For a random variable (for example, the score of a player who uses random guessing), the expected value is the sum of all values of this variable, weighted with their respective probabilities. A player using random guessing selects an arbitrary image out of the n currently displayed images. As only one image at a time is the goal, the probability of guessing correctly is $P_r = \frac{1}{n}$. The probability of being wrong is therefore $P_w = 1 - \frac{1}{n} = \frac{n-1}{n}$. Then for any single guess, the expected value of the Guesser's (and consequently the Describer's) score S is $E[S] = P_r \cdot S_r(n) + P_w \cdot S_w(n)$, with $S_r(n)$ and $S_w(n)$ being the score bonus or (negative) penalty for guessing correctly or wrong respectively. Since these values may depend on the number of remaining images, they are modeled as functions of n. By inserting the definition of the probabilities, $E[S] = \frac{1}{n} \cdot S_r + \frac{n-1}{n} \cdot S_w$. It is reasonable to define being "successful" in the game as earning a positive score. The expected value of the score of players using random guessing should therefore be less than or equal zero:

$$\begin{aligned} & E[S] \leq 0 \\ \iff & P_r \cdot S_r(n) + P_w \cdot S_w(n) \leq 0 \\ \iff & \frac{1}{n} \cdot S_r(n) \leq -\frac{n-1}{n} \cdot S_w(n) \\ \iff & S_r(n) \leq -(n-1) \cdot S_w(n) \end{aligned} \tag{3.1}$$

Calculating the expected score of a player for an entire game round is more difficult, since n changes depending on the actions of the Guesser. However, there is a simple approach for ensuring that the expected value remains negative. By choosing $S_r(n)$ and $S_w(n)$ in a way that fulfills equation 3.1 for every $n > 1$ (as $n <= 1$ does not occur in the game), each turn of the game will have a negative expected score and thus the entire game will as well. For example, by choosing $S_r(n) = 1$ and $S_w(n) = -1$, equation 3.1 is fulfilled for any $n > 1$.

In most cases, players will not guess randomly from the entire set of n images. By applying the given description, the Guesser can often narrow the set of candidate images down to a smaller number of images. To ensure that random guessing still yields a negative expected score in this situation, the condition imposed by equation 3.1 must be defined more strictly:

$$S_r(n) <= -S_w(n) \tag{3.2}$$

If equation 3.2 is fulfilled, neither random guessing from all images nor random guessing from any subset of the images has a positive expected score. Even random guessing from only a pair of candidate images can be expected to yield a negative score.

In Karido, the bonus for guessing correctly increases as the number of remaining images n decreases. This is somewhat counterintuitive, as one would expect the game to become easier when there are fewer images too choose from. Thus, one would expect to gain the most points from the most difficult turns. However, there are several factors that make increasing scores a sensible alternative. Firstly, the duration of a round is limited (see Section 3.2.3). Therefore, players will have to act efficiently to proceed to the later turns of a round in which there are fewer images. Secondly, some images are potentially easier to describe than others (for example, because of a special object depicted in them). As the Describer can select the goal once the previous one was successfully guessed, it is likely that the players will try to remove the easier images first, leaving the difficult ones for the

turns in which there are higher scores. Lastly, increasing scores should motivate players, especially in combination with the limited duration of the round. A scheme of increasing bonuses is therefore a reasonable scoring method in terms of resulting player motivation.

To ensure that the expected scores remain negative, increasing bonuses imply increasing (in terms of absolute value) penalties. In KARIDO, the scoring is defined in the following way: Let m be the one-based index of the current goal image (thus, $m = 1$ at the start of a round, $m = 2$ after one image has been removed, etc.). Then $S_r(n) := 10 \cdot m = 10 \cdot (10 - n)$ and $S_w(n) = -11 \cdot m = -11 \cdot (10 - n)$. Thus, the number of played images is multiplied by a constant value (10 and -11 respectively) to get $S_r(n)$ and $S_w(n)$. These constant factors will be referred to as *bonus factor* and *penalty factor*. The penalty factor has been chosen to be slightly larger than the bonus factor in terms of its absolute value. The reasoning behind this decision is to avoid that all resulting scores are multiples of 10, resulting in more interesting comparisons between players (for example, in a high-score list).

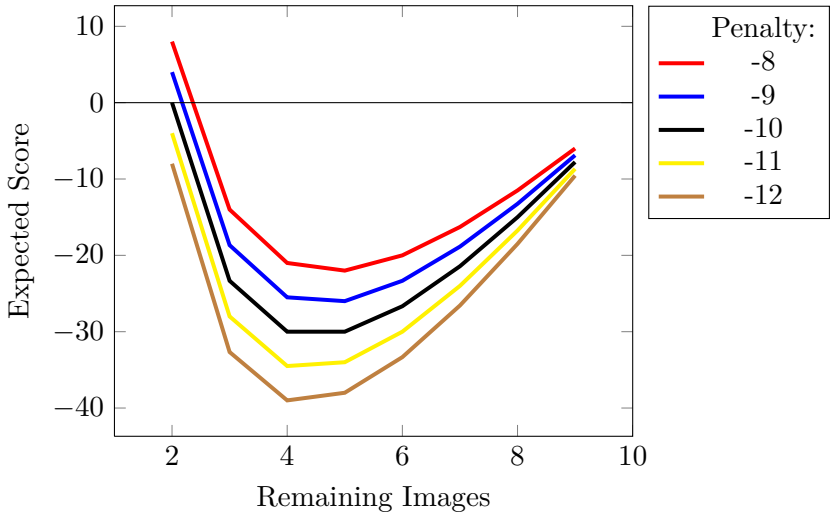


Figure 8: Expected score per turn of a player using random guessing in KARIDO, as a function of the number of remaining images n and the penalty factor for guessing wrong. The bonus factor is fixed to 10. Although the expected value is a function of an integer argument, linear interpolations between the values have been added for visual clarity.

In this scheme, the scores and penalties grow linearly with the decreasing number of remaining images n. In contrast, the probability of guessing correctly is inversely proportional to the number of remaining images. Thus, the expected score $E[S](n)$ is not constant in n. Figure 8 illustrates this behavior. Each plot in the figure represents the expected score as a function of the number of remaining images n, for a given penalty factor. The bonus factor is fixed to 10 as used in the implementation of KARIDO. The plot shows that the expected score is highest at the beginning and end of each round and lower in between.

By using a different scoring scheme, for example based on the probability of guessing correctly, the expected score could be made constant. However, this would result in real-valued scores or at least scores which are not round numbers, which are likely less appealing

to the players. Furthermore, the non-constant expected value can even be considered an advantage. In the beginning of a round, the players are not accustomed to their teammate and are more likely to guess wrong. By imposing a lesser penalty, early frustration can be avoided. Similarly, at the final turn of a game, it is reasonable not to spoil the feeling of achievement of the players by applying large penalties.

The two uppermost lines in Figure 8 show the expected score for a penalty factor of -8 and -9 respectively. The plots show that the expected score in the final turn of a game becomes positive for penalty factors above -10. To ensure that the expected score remains negative, a penalty factor of -11 has been chosen for KARIDO.

3.3. Potential Issues

KARIDO was designed with the goal of improving on existing image labeling games. Nevertheless, the new game design is not without issues. In part, these issues are introduced by design decisions for the game, such as the relatively unrestricted communication. Other problems, such a cheating, arise in almost all GWAPs. The following Section of this thesis gives an overview of both types of issues and their proposed solutions.

3.3.1. Cheating

As established before, it is impossible to circumvent the verification method used by KARIDO using only textual communication. However, the possibility of communicating the content of images directly can be used to cheat. There are several ways in which such communication could be achieved.

- KARIDO is a Web-based game. This means that the images displayed to the players are downloaded from a specific Web address by each player's browser software. It is easily possible for any Describer with minimal technical knowledge to find out the address of the selected image. This address could then be sent to the Guesser, who could compare it to the addresses of the images and select the right image. This can be avoided in several ways. Player input could be filtered (for example to allow only words from a dictionary) to prevent players from sending Web addresses. In KARIDO the issue is solved through address obfuscation for images. The addresses of the images are random strings (independently created for each player) that can only be resolved by the central game server. Thus, players cannot match any exchanged addresses.
- Many current instant messaging applications such as Skype [sky] allow users to share the image currently shown on their computer screen with other users. If the players of KARIDO used the textual communication channel to exchange their contact information for an instant messenger, they could use a "screen sharing" feature to cheat. This could also be circumvented by restricting the communication between players to disallow the exchange of contact information. Furthermore, widespread use of this attack seems unlikely, as not all players will be users of the same instant messenger and most players will likely be reluctant to share their contact information with complete strangers.

- Even more sophisticated attacks could be constructed to transfer image information over the textual communications channel. While the Web address of the images are different for each player, the content of the actual files is identical. By creating a hash value for each image and comparing the hash values, the players could communicate the goal image directly. This process could be automated completely. As before, this attack could be prevented by using input filtering. Furthermore, the attack requires significant effort and two cooperating players. Thus, this attack can be made highly unlikely by random matching of the players.

To conclude, the most likely attack scenarios for KARIDO can be suppressed by restricting the communication between the players. However, as the attacks seem relatively unlikely even for unrestricted communication, no filtering is performed until the use of such attacks becomes evident.

3.3.2. Inappropriate Player Communication

Any public service (such as a game, a online chat room or a Web forum) that allows communication between its users has to consider whether this communication needs to be restricted. This is especially important for systems in which the input of the users is persistent and public, such as Web forums. Such systems can easily be abused, for example to spread racist viewpoints or insults. In some jurisdictions, the operator of a service used to publish such content can be held liable and face severe penalties. Such considerations are usually less important for games in which the communication is usually shared only between few players and cannot be accessed after a game has ended. Nevertheless, if communication is allowed between players, there is a risk that players will use these channels to send inappropriate messages. Especially for games, which by their very nature are likely to attract a young audience, it is important to protect minors or – if such protection is not possible or not wanted – to restrict the access to the game to adults.

Output-agreement games such as the ESP GAME handle player communication very elegantly. As no direct communication between the players is possible, any participant of the game can only see her partner's input if she has entered the same data. Thus, these games are completely safe for users of all ages (assuming the image database contains only family-friendly material).

In contrast, KARIDO uses unrestricted communication between the players. Consequently, access to the game should not be possible for minors. Currently, neither filtering of player communication nor access restrictions are implemented. As with the measures against cheating, the communication between players will be monitored. If inappropriate communication should occur, measures can be taken.

4. Implementation of KARIDO

While the previous Section describes the functional design of the proposed new GWAP KARIDO, this Section gives a more technical overview of the implementation of the game.

4.1. ARTIGO Base Implementation

From the very beginning, KARIDO was planned to be based on the so-called ARTIGO 4.0 platform. Artigo [art] is a GWAP designed by a team around Hubertus Kohle at the University of Munich. The functional goal of the game is to label historical artworks using the same game mechanics as the ESP GAME. The main difference between the games consists of the different image databases. At the time of writing, version 3.0 of the game is publicly accessible. Version 4.0 of the game is a completely new implementation, currently undergoing beta-testing.

The primary goal of the reimplementation is to create an extensible framework, allowing the creation of new games without having to rewrite any base functionality. To achieve this goal, Artigo 4.0 relies on the *Model-View-Controller* paradigm. This programming paradigm separates the concerns in an application. The *model* describes the data objects used by the application. The *controller* manipulates these objects and thus describes the functionality provided by an application. Lastly, the *view* represents the user interface. These three components of an application are loosely coupled and thus can be exchanged individually without modifying the other components.

The ARTIGO framework is primarily designed to support the creation of labeling games. Its data model provides a number of fundamental entities:

- A *Person* represents a registered or anonymous player. The framework also handles registration and authentication of players.
- A *GameType* uniquely identifies a game implemented using the ARTIGO framework. This allows the coexistence of an arbitrary number of games in the same environment. The game type also defines basic parameters of a game, such as round duration or number of players.
- A *GameSession* has a GameType and contains an arbitrary number of GameRounds.
- A *GameRound* holds a reference to the Person who has played this round. Thus, for a round of a game with two players, two GameRounds are added to the corresponding game session. Additionally, the GameRound holds the Actions performed by its player and the score resulting from these Actions. Finally, a list of Resources that were processed in this round is stored.
- A *Resource* is an abstract superclass from which concrete Resources can be derived. These concrete resources are the entities for which labels are to be collected (for example, a piece of art).
- An *ArtResource* represents a historical artwork that is to be tagged in ARTIGO. It holds a reference to its artist, titles in several languages and a location from which the actual image can be retrieved.

- An *Action* is the superclass of all actions a player of the game can take.
- A *Tagging* is a subclass of Action and describes the assignment of a Tag to a Resource by a Person, regardless of the validity of the Tag.
- A *Tag* represents a unique label. This label can be assigned to an arbitrary number of resources using Taggings. For example, if three players assign the label "cat" to an image, three Taggings are created, all of which reference the same Tag. A Tag holds a textual representation of its label and a designator for its language. This language designator is assigned based on the language selected by the user who created the tag.

4.2. Seam Web Framework

Seam [sea] is a Web application framework. It unifies several lower-level frameworks[8] and is designed to facilitate development of applications using the MVC paradigm.

Seam uses so called *components* to represent the objects that comprise the Controller and the Model. These components can be created explicitly or automatically when they are needed. Furthermore, the components can interact while still remaining exchangeable through the use of *bijection*. Each component can mark other components with which it has to interact to achieve its functionality. These other components can be *injected* into the current component as well as *outjected* (or both). Injection allows a component to access other components, whereas outjection allows a component to publish its results to other components.

To manage the lifetime and accessibility of these components, Seam uses a system of so called *contexts*. Contexts are defined periods of time during which a component exists. The context in which a component exists is called the component's scope. Seam provides the following contexts:

- The *stateless* context is a special case. Seam allows the creation of components that do not carry state information and thus only exist while one of their methods is executed. These components can be marked to belong to the stateless context. Thus, the stateless context is only used as a marker and does not actually store components.
- The *event* context is the one with the shortest lifetime. As Seam is a Web application framework, it has to deal with a number of Web-specific concepts. The event context is an abstraction of a Web request, i.e., the time between a query sent by a user's Web browser and the response of the server.
- The *page* context is active as long as a single page of the application is viewed in a user's browser. This includes any requests originating from this page.
- The *conversation* context encapsulates a task the user of an application wants to complete. A conversation is started by the application and closed by it, once the given task has been completed.
- The *session* context is an abstraction of the concept of a Web session. This context remains valid as long as a user remains logged in to the application. For users that are not logged in, the context remains active as long as the session is maintained by

[8] Primarily, Enterprise Java Beans (EJB) and JavaServer Faces

their Web browser.

- The *business process* context encapsulates *business processes*, i.e. complex and potentially very long interactions between multiple users. These processes are managed by a business process management engine.
- The *application* context is active over the entire lifetime of an application. In this case, "lifetime" means the timespan over which an application remains deployed on a server. The components in all other contexts can only be accessed by their individual users. In contrast, application-scoped components can be accessed by all users of the system.

The developer of a SEAM application can define in which context a component resides. This enables fine-grained control of access rights and enables SEAM to automatically manage the lifetime of the components. Additionally, SEAM makes it possible to deploy an application to several *clustered* servers. This means that several connected servers execute the same application and automatically maintain a synchronized state. This allows to maintain scalability with growing numbers of users and increases the reliability of an application.

4.3. Special Challenges in the Implementation of KARIDO

KARIDO has been implemented using the SEAM and ARTIGO frameworks described above. An additional framework was created to support concurrent interaction of several players (see sections 4.3.1 and 4.4). Much of the implementation of KARIDO has been relatively straight-forward. The underlying game mechanics are comparatively simple and were easy to transform into working code.

Other parts of the implementation proved to be more challenging. The development of the Web-based user interface was one of the larger challenges. Using interface components provided by SEAM (such as text boxes) is simple. However, building a Web site and custom components (such as the image grid) that look identical on most browsers and remain responsive even on older hardware turned out to be more difficult than expected. These problems were solved primarily using trial and error and the resulting solutions reflect the peculiarities of common Web-browsers. Therefore, the implementation of the user interface is not described in detail.

In other instances, the conventions imposed by SEAM and the ARTIGO base implementation made it necessary to use the frameworks in ways that were not intended by their creators (see for example the following Section 4.3.1). Other parts of the implementation were challenging as a result of the underlying problems (for example, simulating a human player in a believable manner, see Section 4.3.2). The following Section of this thesis describes the challenges encountered during the implementation of KARIDO and the solutions or workarounds found for each problem.

4.3.1. Using SEAM for Real-Time Interaction

One of the fundamental requirements for most Games With A Purpose is two player interaction. In KARIDO, the players exchange descriptions and questions and various information

must be synchronized, such as the current score, the remaining time or the grid of images. The actions taken by the players can occur in rapid succession and as little delay as possible should be introduced by the system.

Games in which both players can take actions asynchronously are commonly called "real-time" games. The opposite are "turn-based" games, in which the players perform their actions in a strictly defined succession. In turn-based games, the latency introduced by the system is of little importance (consider, as a prime example, correspondence chess, in which two players exchange their moves by mail). In contrast, real-time games impose far stricter requirements on latency. The actual latency at which a game can be considered unplayable depends on the design of the game and personal preferences. For action games, the latency should ideally be below 50ms, with 150ms often considered an upper limit, as stated by Borella [2]. Karido implements both a turn-based and a real-time mode. However, even for the real-time mode, the requirements on latency are less strict than for action games. A delay of a second is easily acceptable for all modes of Karido.

An issue arises from the fact that Karido is Web-based. The Hypertext Transfer Protocol used by the Web is stateless and – in a sense – unidirectional. The client (usually a Web browser), sends a request to the server, which responds (usually with the content of the request Web page) and closes the connection. Thus, before the invention of asynchronous requests, dynamic elements of a page either had to be calculated locally or required a complete reload of the page. Recently, a technology called *AJAX* made it possible to dynamically reload parts of a Web page. However, such reloads still have to be triggered by the client – thus, the server cannot notify the client of any changes (such as the arrival of a new description). There are experimental approaches to allow Web servers to notify clients, but at the time of writing, there is no standard method for achieving this. To solve this issue, the RichFaces library included with Seam supports polling. This means that the client sends light-weight requests to the server in regular intervals and is then informed of any changes. To avoid excessive network traffic, these intervals should be relatively long. Karido uses an interval of 500ms, which appears to be an acceptable trade-off between network load and game latency.

An additional issue arises from Seam's contextual model. As described above, the basic contexts of the Seam framework only allow access to their components by a single user. The *business process* context relies on a management engine to coordinate the access to components by several users. Business processes impose strict requirements on reliability and persistence (for example, a business process should be continued even if the application is moved to a different server). Therefore, all interactions between the users of a business process are persisted into a database. Because of the large number of quick interactions in a GWAP, the database could become a bottleneck.

The application context provided by Seam allows all users to access its components. However, it is not possible to restrict or coordinate the access in a way that allows *pairs* of users to possess shared data. It is therefore necessary to implement a different way of handling data synchronization between players. A framework allowing real-time multi-player interaction has been created for Karido. The framework is designed to be extensible and can be used to implement arbitrary multi-player games.

The basic principle of the framework is to provide a common entry point for players

requesting a game session. The component that provides this entry point resides in the application context and can therefore coordinate the games for all players. This central component handles queuing and matching of players and creates components that are shared between the players. A reference to the shared component can be outjected into the conversation context of each user, allowing players to access the shared data.

Section 4.4 gives a more detailed overview of the multi-player framework. It describes the classes provided by the framework and their respective functions and gives a short guide on how to implement a new game using the framework.

4.3.2. Implementing Single Player Games

As described in Section 2.2.1, all GWAPs that only allow a specific number of players to collaborate in a game round should enable players to play alone if no partners are available. Thus, the same holds true for Karido. As Karido is a inversion-problem game, it is necessary to simulate a player that interacts with the human player. This simulated player will be refered to as a *bot*. This bot must be able to assume both the roles of Guesser and Describer and ideally should behave indistinguishable from a human player.

Describer Bot. The Describer bot in Karido must fulfill two tasks: Firstly, it needs to describe the current goal image. Secondly, it must answer any questions posed by the Guesser.

The first task can be achieved by replaying rounds, similar to most other GWAPs that provide a single player mode. In other GWAPs, the round to be replayed is selected based on the current resource that should be described. However, Karido uses a set of nine images. Selecting a round to replay solely based on the current goal resource is unlikely to yield good results, as the description of the goal image largely depends on the other images currently shown. Selecting a round in which all nine current images were played is unfeasible, as the probability that such a round exists is relatively low.

Therefore, the first step in simulating a Describer is to randomly select a previously played round. Only rounds in which more than six labels were assigned are considered in this selection. The images played in the previous round are used in the current round. The tags of the first goal image are then replayed using the same delays as in the recorded round. Once the right image has been selected, the simulated Describer chooses a new image. The tags for this image are replayed. To ensure that all tags are replayed in the proper context, the simulated Describer selects the same sequence of goal images that was played in the previous round.

The delays used when replaying the tags are relative to the time at which the current goal image was selected. This is illustrated by Figure 9. The upper time-line shows the progress of the recorded round: The Describer has sent two tags before the Guesser selected the right image. A new image is selected and Describer sends a third tag. The lower timeline shows the timing of the same round being replayed. The Describer bot sends the first recorded tag. The human Guesser selects the right image quicker than the Guesser in the recorded round. Thus, the second tag is discarded. The simulated Describer selects the next image.

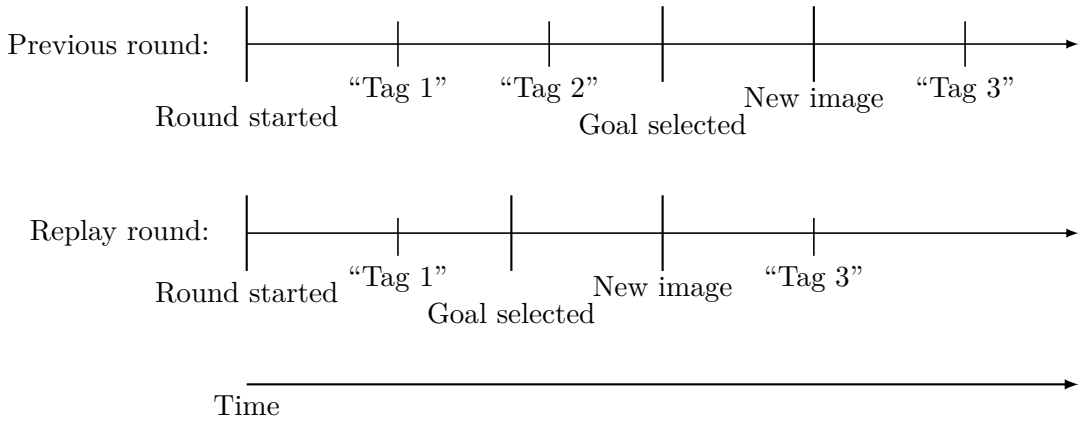


Figure 9: Illustration of the timing used when replaying rounds in KARIDO.

Tag three is sent using the same delay between the selection of the new image and the sending of the tag as in the recorded round.

In the example described above, the Guesser in the replayed round is quicker than the Guesser in the original recorded round. In this case, the bot can proceed by dropping any remaining tags for the selected resource and selecting a new image. In contrast, if the human Guesser takes longer than the Guesser in the original round to select the correct image, there are no further tags that can be replayed. If the simulated Describer suddenly stops sending tags, the illusion of playing with an actual partner is destroyed. To avoid this, the simulated Describer selects random tags and sends these tags with randomized delays once no more recorded tags are available. This sudden change in description behavior is potentially noticeable. In most cases however, the Guesser should be able to select the right image (as this was also achieved by the Guesser in the recorded round) in time, thus reducing the probability that randomly selected tags have to be played.

The described fall-back solution fails for images that are not yet tagged. This is a fundamental problem in any GWAP that allows single player games using simulated partners. However, as long as at least two players use the game regularly, simulating a player without destroying the illusion of a human player can be successful. In the first session of the game, it is not possible to simulate the Describer. However, the tags entered by the player are recorded. When a second round is played by a different player, this round can be replayed. Additionally, another round is recorded. This round can be replayed to another player. Thus, each game session records a new round and verifies the labels created in a previous round.

In addition to sending descriptions, a simulated Describer must be able to answer questions posed by the human Guesser. These questions are equivalent to tags of the Describer, but instead of implying an assertion ("The goal image depicts a tree.") they imply an inquiry ("Does the goal image depict a tree?"). Recorded rounds cannot be used to answer questions, as it is relatively unlikely that two independent Guessers pose the same question. Therefore, all tags assigned to an image are used to answer questions. If the content of a question has ever been used to describe an image, the question is answered positively. If no label with the content of the question exists, the question is answered negatively. As

players often spell the same word differently and as the set of labels for an image is usually incomplete (especially for images that have not been played often), falsely negative answers are a greater issue than falsely positive ones. Therefore, all tags assigned to an image are used for answering questions, even if they have not yet been verified.

Guesser Bot. Like the Describer bot, the Guesser bot in KARIDO must fulfill two tasks: Firstly, it must interpret the descriptions given by the Describer and select an image once it is reasonably certain that it is the goal image. Secondly, if several potential images remain, it should ask questions to reduce the number of candidate images. For both tasks, the bot relies on the tags assigned to the images.

As described in Section 2.2.2, the behavior of the simulated Guesser does not directly influence the quality of the data collected in the game, because all tags are validated by an independent player. However, if players discover that they can score points as Describers without entering valid descriptions, no more valid information will be entered into the system and the verified information will stagnate. It is therefore necessary to simulate a Guesser that only selects the goal image if it has been described accurately.

Entered \ Assigned	"tree", "dog"	"tree", "house"	"red", "car", "house"
"tree"	100%	100%	0%
"tree", "house"	50%	100%	50%
"tree", "car", "red"	33.3%	33.3%	66.6%
"tree", "car", –"dog"	0%	50.0%	50.0%
"red", "brick"	0%	0%	50%

Figure 10: The percentage of entered labels assigned to differently tagged resources. This number is used by the Guesser bot to select images. Tags prefixed with a dash and set in red represent questions that were answered negatively.

The following section describes the behavior of the simulated Guesser in textual form. The Guesser bot uses a relatively long chain of decisions to decide which actions to take. The description of this flow is somewhat convoluted by very nature of the subject. Figure 11 illustrates the same flow graphically and should be considered alongside the following text.

As a first step, the percentage of entered tags assigned to the images in the grid is calculated. This value will from now on be called the *match percentage* of an image. The calculation is illustrated in Figure 10. The columns represent three images along with their assigned tags. Each line represents a different set of tags entered by the human Describer. The cells of the table contain the calculated match percentages. All images for which at least one tag matches (and which have not yet been wrongly guessed) are sorted by decreasing match percentage. The resulting list of images is used to decide which image to select. The first image from the list is selected if:

- the list contains only one image (see the bottom line of Figure 10).
- it is the only image to which all entered tags have been applied (see the second line

of Figure 10).

- its match percentage is at least 1.2 times larger than one of the next image (see the third line of Figure 10).
- more than n descriptions have been created by the Describer. n is randomly selected to fulfill $2 \leq n \leq 4$. This heuristic ensures that the Guesser bot eventually starts to guess randomly if the Describer fails to refine the description.

If none of these criteria are applicable, the simulated Guesser prepares to ask a question. All images which have at least half the match percentage of the first image are added to a second list. This list of candidate images is used to ask a question (see below).

It is also possible that there are no images to which any of the entered tags apply. In this case, the bot falls back to random guessing. If more than n descriptions have been entered by the Describer (n is randomly selected to fulfill $0 \leq n \leq 2$), a guess is taken. With a probability of 70%, the right image is selected immediately. Alternatively, a random image is selected and guessed. If the number of entered tags is less than or equal to n, a random image is selected and added to the list of candidate resources for asking questions.

If the current goal image has less than five tags assigned to it, its match percentage is considered unreliable. In this case, an alternative selection is performed. If there are other images which have a non-zero match percentage and if the highest match percentage is larger than p, the corresponding image is selected. p is chosen randomly between 30% and 130%. This means that if the entered tags accurately describe an image other than the goal image, it is likely that this image is selected. If the highest match percentage is less than or equal to p, the goal image is selected.

If no other images share any tags with the entered description, the following approach is taken. If more than m descriptions have been entered (with m randomly selected to be between 0 and 2), a guess is taken. With a probability of 80%, the right image is selected immediately. This is reasonable, because the fact that no image shares any tags with the description implies that either the description is wrong or very uncommon or that the description is only accurate for the goal image. In case of a wrong description, little harm is done. In case the description is accurate, it is desirable to reward the player by guessing correctly. If the right image is not selected immediately, a random image is chosen and guessed. Finally, if no more than m descriptions have been entered, a random image is chosen and added to the list of candidate resources for asking questions.

If no image is selected and guessed in the steps described above (see the first line of Figure 10), a list of candidate resources has been assembled. This list contains the images which the simulated Guesser currently considers candidates for the goal image. Questions should be formulated to reject as many images as possible when answered. To achieve this, the simulated Guesser selects the first image from the list of candidates. The list of all tags assigned to this image is generated. All tags which have been applied to any of the remaining candidate images are discarded. The tags that are unique for the selected image remain. This list is ordered by number of times each tag has been applied. One tag is randomly selected from the three most common labels and sent to the Describer as a question. As the tag is unique to one of the candidate images, if the question is answered negatively, only one image can be removed from the list of candidates. However, if the question is answered

positively, the respective image can be assumed to be the goal image with a high probability.

Lastly, after any posed questions have been answered, these answers must be used to reduce the number of candidate images. All questions which were answered positively are treated like tags created by the Describer. They are directly used to select the candidate resources and contribute to the calculated match percentage of the images. In contrast, questions which were answered negatively are added to a list of blocked tags. For each candidate image, the number of blocked tags assigned to this image is calculated. This value is subtracted from the number of matching tags used in calculating the match percentage of the images (see the fourth line of Figure 10). Overall, each question is equivalent to one tag added by the Describer (in case of a positive answer) or invalidates one such tag (in case of a negative answer).

Timing. For both simulated Guesser and Describer it is important to maintain a quasi-human behavior. This means taking into account the time it takes for a human player to decide which action to perform and to perform the action. When replaying previous rounds, the Describer bot can rely on the recorded delays to provide realistic timing. One question is answered immediately each time the AI is active. The reasoning behind this decision is that the delay introduced by the Web-based user interface masks the immediate answer and that the AI is only active once every two seconds. Additionally, as answering a question takes only a single click of the mouse, human players can be expected to be quick at answering a question as well. Furthermore, the Describer is focused on the goal image and should therefore be able to answer questions without first having to look at the image.

In contrast, formulating descriptions or questions and then typing them takes more time. To model this delay, the bot uses several mechanisms. A Describer's primary task is to add descriptions. As such, the Describer can be expected to be adding descriptions more or less constantly. Thus, the delay introduced by typing the tags significantly influences the overall delay. When sending random tags, the delay in milliseconds is calculated as $1000 + l \cdot (200 + rand(300)) + rand(2000)$, with l representing the number of characters in the sent tag and the $rand(x)$ function returning a random value between 0 and x.

The primary task of a Guesser is to select the appropriate goal image based on the given description. As such, asking questions is only necessary if the Describer fails to accurately describe the goal image or if the game is played in turn-based mode. Thus, questions are primarily asked at a point at which the pace of the game is slow. A Guesser posing a question is therefore likely to put more thought into the question then a Describer sending a description. As a result, typing speed has less influence on the overall delay. Therefore, the delay introduced by posing a question is defined as $1000 + rand(6000)$ milliseconds.

4.3.3. Selecting Images by Similarity

A fundamental element of KARIDO is the selection of images to be displayed to the players. These images should be selected by similarity. Additionally, the selection process should have a random component to ensure that the players get different images in each round. Finally, the selection method should encourage an equal number of tags for all images in the database. This means that images with few labels should be preferred in the selection.

The following approach is used in KARIDO: A *base image* is selected at random from the 100 least tagged images in the database. This ensures that eventually all images in the database have a similar number of tags. The selected base image is set as the first goal image. Next, additional images with similar tags as the base image are selected. This similarity between two images is measured as the number of times that any tag has been applied to both images. As a result, the more often a unique tag has been applied, the higher its weight in the selection process. To reduce the probability that a player is given the same set of images multiple times, the 36 most similar images are selected. From this list, eight images are chosen at random to fill the grid of nine images.

It is possible that less than nine images have been found after the process described above is completed. For example, if the selected base resource is not tagged at all, no similar resources can be found. Alternatively, if few images in the database are tagged (for example, shortly after the game has just been published), too few resources might have been found. In this event, KARIDO adds randomly selected resources to fill the image grid.

4.3.4. Ensuring Scalable Database Access

The ARTIGO framework and game as well as KARIDO are designed with large image databases in mind ("large" meaning hundreds of thousands of images). This in turn means that a large number of game rounds and tags must be handled by the game. The image selection described above requires complex queries to the database. For example, selecting the least tagged images means calculating the number of labels assigned to each image. The labels are assigned through Taggings, which hold a reference to a tag and a resource to which this tag has been applied. Thus, to count the number of tags assigned to an image, all Taggings (which means potentially millions of them) have to be scanned and tested whether they point to a given image.

The selection of similar images presents a performance bottleneck as well. All Taggings have to be scanned to select the tags which have been applied to the base image. Then, all Taggings must be scanned again to find out which other resources have been assigned the same labels. These Taggings must finally be counted for each image.

These database queries have to be performed several times for each game session. The requests should be completed with minimal delay to avoid that players have to wait for the game to load and that database access becomes a bottleneck when larger numbers of users access the game. Despite several attempts to optimize the database requests, no satisfactory performance could be achieved. Even if such optimization was possible, the performance might still be insufficient in the future when more images and tags are added to the database.

To avoid this problem, KARIDO limits the number of images which are considered for these complicated requests. Before the request is performed, 1000 images are randomly selected from the database (which can be achieved in very little time, regardless of the size of the database). The query is then performed considering only the selected 1000 images. As a result, many properties of the selected images are not guaranteed to be valid. For example, the base image is selected from the least tagged *preselected* images, as opposed to the least tagged images in the entire database. Thus, it is possible (but unlikely) that an

image without any tags is never selected as the base image. The same holds true for the selected similar images. It is possible (but unlikely) that only images that are not similar to the base image are selected.

However, the preselected images are a random sample of the entire database and can therefore be expected to exhibit the same properties as the entirety of the images (similar to random samples used in polls). Additionally, the sampling is repeated two times in each game session. Thus, the described restriction of database queries will not affect the overall output of the game as described in Section 3.

4.4. Documentation of the Framework Developed for KARIDO

As described in Section 4.3.1, a custom framework was built to support real-time player interaction in KARIDO. This Section describes the base classes of the framework and how they can be used to implement custom games.

4.4.1. Classes Provided by the Framework

The interaction framework provides three base classes that contain the basic functionality necessary for implementing real-time games. All three classes are *generic* and *abstract*. This means that they must be extended to create actual games. The subclasses which extend the base classes contain all additional code that describes the behavior of the game. As only the core functionality is provided by the base classes, the framework makes it easy to implement a wide range of different game designs. This Section gives an overview of the functionality provided by the base classes.

PlayerMatcher The most important class in the multi-player interaction framework is the `PlayerMatcher`. It lies within the application context (see Section 4.2) and represents the global entry point for all users wanting to participate in a game. The most important method provided by the `PlayerMatcher` is `match`. The method returns a `SharedGame` and a `Player` object. It can be called without parameters. In this case, it creates a `Player` and a `SharedGame` using default parameters. Alternatively, a `Player` object and a `SharedGame` can be provided by the caller. This makes it possible to customize the parameters of the `SharedGame` as opposed to relying on the default value. For example, the two variants of KARIDO rely on the same set of classes, but customize the parameters of the `SharedGame` to implement the different game mechanics.

In the `match` method, the `PlayerMatcher` searches the maintained list of running games to ensure that no player is able to participate in multiple games at the same time. If an existing game is found and it is *compatible* (see the next paragraph for a definition) with the game provided by the caller, the existing `SharedGame` and the existing `Player` object are returned. This means that the player is redirected to the existing game. In contrast, if the existing game is not compatible with the requested new game, the existing game is terminated and the new `SharedGame` is used to find a partner for the player.

As the next step, the `PlayerMatcher` searches the list of waiting players. If an entry

for the player already exists, this entry is removed to ensure that no player can request to play two games at the same time. If an entry for another player exists and the `SharedGame` owned by this player is compatible with the `SharedGame` provided by the caller of the `match` method, the two players are matched.

If no such entry exists, the `Player` object is added to the queue of waiting players, along with the `SharedGame`. The `SharedGame` and the `Player` object are returned and can then be outjected into the context of the caller.

When two players are matched, the `SharedGame` of the player who has first joined the game becomes the common game of both players. The `SharedGame` provided by the second player is discarded. This is necessary because the first `SharedGame` has already been outjected into the context of the first player. The `Player` objects are added to the `SharedGame` and each `Player` object is given a reference to its partner. ID numbers are assigned to each `Player`, which can later be used to determine their roles in asymmetric games. A notification is added to both `Player` objects signaling the recent match. Finally, the `newRound` method of the `SharedGame` is called to initiate the first round of the game.

If a `Player` object has not been matched after a given timeout (which can be configured in the `Player` class), it is matched with an *AI*-player (unless such matching was disabled). AI stands for "artificial intelligence" and represents a program designed to substitute a human player in the game (see Section 4.3.2 for an example). The `Ai` class must be provided by the creator of a game and is a subclass of the `Player` subclass of the given game. As such, it can be used to simulate the behavior of a human player in whatever way necessary in a specific game.

SharedGame The `SharedGame` class encapsulates all data that is shared between the players of a game. In the base class, this data primarily consists of information necessary for running a game – such as the index of the current round, the total number of rounds, the duration of each round or the score achieved by the players. The methods provided by the `SharedGame` base class consists primarily of accessors for the shared properties. Two abstract methods must be implemented by the subclasses created for a game. The `startNewRound` method is called each time a new round of the game has begun and can be used to initialize any game specific properties. In KARIDO, for example, it is used to select a new set of images.

The `isCompatible` method expects a `SharedGame` as parameter and returns a boolean value indicating whether the two games are compatible. This method is used by the `PlayerMatcher` to determine whether two players can be matched. It is similar to the `equals` method provided by all Java objects. The `isCompatible` method is necessary because `SharedGame` subclasses can contain an arbitrary number of parameters and only the creator of the game knows which attributes make games incompatible. For example, KARIDO enforces that only players using the same language can play together. Additionally, only games of the same type (meaning turn-based or real-time) are compatible.

Additionally, the `SharedGame` holds a component that manages the `GameSession`. The `GameSession` is provided by the ARTIGO framework. While the `GameRound` objects are managed for each player individually, all players participating in a game session share the

same `GameSession` object in which their `GameRound` objects are collected.

Player The `Player` class represents a player of a game. Each `Player` contains a `Person` attribute. The `Person` class is provided by the ARTIGO framework and represents a person that is logged into the system or plays anonymously. The `Player` class also holds a `GameRound` object, which represents a game round as defined by the ARTIGO framework. This object can, for example, be used to append custom actions to the game round.

Each `Player` is assigned a list of *notifiers*. These notifiers are text strings, which can be added using the `addNotifier` and `signal` methods. The notifiers are used by the `Player` object as well as the user interface to react to events in a game. For example, once a player has sent a tag to her partner in KARIDO, a notifier "tags" is added. This notifier is intercepted by the polling method of the partner's user interface. The user interface responds by displaying the updated list of tags.

The `addNotifier` and `addNotifierAll` methods append a string to the list of notifiers of one or all `Player`s respectively. In contrast, the `signal` method additionally sets a flag in the `PushEventListener` held by each `Player`. This flag is polled by the Web browser of a player. If the flag is set, the list of notifiers is retrieved. The user interface then polls and updates any data sources for which a notifier has been set. Thus, instead of constantly polling all data sources, only a single flag has to be polled. This reduces the network traffic generated by the polling and allows shorter polling intervals which in turn lead to a more responsive user interface.

A central method of the `Player` object is the `poll` method. This method is called in regular intervals (in KARIDO every two seconds) by the player's Web browser. In each call, the time at which this call has taken place is stored in the `Player` object. This is used to detect when a player has disconnected or navigated away from the game. The `poll` method checks how much time has passed since the `Player`'s partner's `poll` method was last called. If this exceeds a given threshold (ten seconds in the current implementation), the player is redirected to a page stating that their partner has left the game.

Additionally, the `poll` method is used in conjunction with the `PlayerMatcher`. As long as no partner has been assigned to the `Player`, the time elapsed since the creation of the `Player` object is measured. If this time exceeds a threshold (six seconds in KARIDO), the `MatchTimeout` method of the `PlayerMatcher` is called. This method creates an `Ai` object and assigns it to the `Player`.

`Ai` objects should not only react to actions taken by the player, but should also be able to take actions at any time. This can be achieved by overriding the `poll` method. As the method is called regularly, it can be used to trigger actions, for example by using timeouts and randomized delays.

The `Player` object defines the abstract method `notified`. This method is polled regularly by the player's Web browser. If the string returned by the method is not `null`, then the player's Web browser is redirected to the corresponding page. For example, this can be used to redirect the players to the scoring page after a game session has been completed.

4.4.2. How to Create a Minimalistic Game

The following Section of the thesis gives a short overview of the creation of a new game using the multi-player framework. As a prerequisite, a working copy of the ARTIGO framework should exists and be fully operational. All additional steps necessary to implement a minimalistic multi-player "game" are described in the following Section.

The implemented "game" is a simplistic chat application for two users. A text box into which messages can be entered is shown to each user. When a player submits a message, it is added to a list of messages that is visible to both players. Single-player operation is supported. The bot's only function is to send the same message over and over in a regular interval.

Preparations. The first step in the implementation of a new game is to create the necessary folder structure, representing the Java namespace of the new game. In this example, the used namespace is `gwap.game.test`. Four new classes must be added into this namespace: `Ai.java`, `Player.java`, `PlayerMatcher.java`, `SharedGame.java`. These classes extend the base classes provided by the framework. The following paragraphs outline the steps necessary to create the new subclasses. The entire source code of these classes can be found in Appendix A.1.

Furthermore, a `GameType` for the new game must be created in the database of the system. For the example, this can be achieved by executing the following SQL command:

```
INSERT INTO gametype
(id, description, label, name, players, roundduration, rounds, workflow,
    platform)
VALUES
(3, 'A simple test scenario', 'Test', 'gwapGameTest', 2, 120, 1, NULL, NULL);
```

Please make sure to choose an `id` that is not yet taken. Longer values for `roundduration` can be set if desired.

PlayerMatcher Class. Even if no special functionality (such as customizable game types) is required, a subclass of `PlayerMatcher` must be created to outject the `Player` and `SharedGame` objects. This subclass is declared as:

```
public class PlayerMatcher extends gwap.game.PlayerMatcher<SharedGame, Player>
```

It is somewhat confusing that the new subclass has the same name as the base class. For example, it would be more straightforward to call it `PlayerMatcherTest`. However, note that the class resides in the namespace `gwap.game.test`. The above declaration is therefore equivalent to:

```
public class gwap.game.test.PlayerMatcher extends
gwap.game.PlayerMatcher<gwap.game.test.SharedGame, gwap.game.test.Player>
```

Thus, by using the same class names for the subclasses in all games, the same declaration can be used for the components. Nevertheless, the proper class names are inserted into the generic parameters.

To allow the `PlayerMatcher` to instantiate various components of the game by name, the implementor must abide by a certain naming scheme. The SEAM name of all components of a game must consist of an arbitrary prefix and their base class name. For example, let the prefix for the example game be `gwapGameTest` (as this was used in the `GameType` in the database). Then the `PlayerMatcher` subclass must be named `gwapGameTestPlayerMatcher` by using the `@Name` annotation. The chosen prefix must be also be stored in the new `PlayerMatcher`. This is done in the constructor by calling the constructor of the base class:

```
public PlayerMatcher() {
    super("gwapGameTest");
}
```

Finally, the following method is the defined entry point for all players of the new game:

```
public void enqueue() {
  Pair<SharedGame, Player> p = match();
  gwapGameTestSharedGame = p.a;
  gwapGameTestPlayer = p.b;
}
```

The `match` method is defined by the `PlayerMatcher` base class and returns a `SharedGame` and `Player` object. These are outjected into the current user's conversation context by the `PlayerMatcher` class.

SharedGame Class. The `SharedGame` class holds all data that is shared by both players. In the example, this data consists of the messages sent by the players. The `SharedGame` subclass holds these messages in a `List` of `String`s. The implemented `sendMessage` method adds a `String` object to the list of messages.

Additionally, a `SharedGame` must implement the `isCompatible` method. The example uses the following method:

```
@Override
public boolean isCompatible(gwap.game.SharedGame game) {
  return (game instanceof SharedGame);
}
```

This method checks whether the passed `SharedGame` is an instance of the `ShareGame` subclass created for the test game. Note that the parameter uses the fully qualified name `gwap.game.SharedGame`. In contrast, the `SharedGame` used in the return statement implicitly refers to the `gwap.game.test.SharedGame` subclass.

Finally, all `SharedGame` subclasses must implement the `startNewRound` method. The example game does not use game rounds (even though `GameRound` objects are automatically created by the framework). Therefore, `startNewRound` is implemented as an empty method.

Player Class. The `Player` class encapsulates all data and methods necessary for a player to interact with the game. In the example, the only data held by the player is a `String` attribute storing the text that is currently entered into the user's text box. The following method is used to send this message:

```
public void sendMessage() {
  gwapGameTestSharedGame.sendMessage(message);
  message = "";
  signalPartner();
```

This method calls the `sendMessage` method of the `SharedGame` to add the message to the common list of messages. It then clears the stored message (and the corresponding text box along with it). Finally, the `signalPartner` method is used to send a notification to the partner. This notification prompts the partner's user interface to update the list of messages.

Additionally, a computer controlled player should later be added to the example game. To check whether a player is allowed to be matched with a simulated player, the `PlayerMatcher` relies on the `allowAi` attribute of the `Player` object. This attribute is initialized to `false`. To enable a player in the example game to be paired with an AI, the following constructor is added to the `Player` subclass:

```
public Player() {
  setAllowAi(true);
}
```

Finally, each `Player` class must override the `notified` method. The `notified` method is called by the user interface to redirect the player to different pages in response to events in the game. The example game uses the following minimal method:

```
  @Override
  public synchronized String notified() {
    if (isNotified("endGame")) {
      return "home";
    }

    return null;
  }
```

The `endGame` notification is sent by the framework if a game session has ended. In response, the `notified` method redirects the user to the home page. In a proper game, the user would usually be redirected to a scoring page.

Ai Class. The creation of an `Ai` class is optional, but the example game illustrates the creation of such a component. The `Ai` class must extend the `Player` subclass created for the game. This ensures that the `PlayerMatcher` can use `Ai` and `Player` objects equivalently.

To ensure proper operation, the `Ai` class must override two basic methods of the `Player` class:

- The `isAi` method is overridden to return `true`. This method is used by the `Player` base class to ensure that the `poll` method of the `Ai` class is called.
- The `isTimedOut` method is overridden to return `false`, as the `Ai` cannot leave the game or be disconnected.

Finally, the example `Ai` class overrides the `poll` method to send messages in regular intervals:

```
@Override
public synchronized String poll(int round) {
  setMessage("This is the AI!");
  sendMessage();
  return "";
}
```

By default, the `poll` method is called every two seconds. Thus, the simulated player sends the message "This is the AI!" every two seconds.

The User Interface. The final step in the implementation of a new game is the creation of the user interface. This is achieved by creating a new Web page (called `testGame.xhtml` in the example). To make this page accessible, a link to it must be placed in the navigational area. For the example game, the following link is used:

```
<s:link view="/testGame.xhtml"
  action="#{gwapGameTestPlayerMatcher.enqueue()}"
  value="Testspiel" />
```

It is essential to specify the `action` attribute correctly. This ensures that the `enqueue` method of the `PlayerMatcher` component is called before the user interface is shown to the player. After the player has been enqueued, she will usually have to wait for a partner. During this time, a message should be displayed. After the player has been matched, the actual game should be shown. This separation between the matching stage and the playing stage can be achieved by using two separate pages or by combining both stages in one page. The example project uses the latter approach.

The player matching stage is handled by the following part of `testGame.xhtml`:

```
<a4j:outputPanel rendered="#{!gwapGameTestPlayer.matched}">
      <ui:include src="widget/waitPlayer.xhtml" />
</a4j:outputPanel>
```

The condition `#{!gwapGameTestPlayer.matched}` results in the content of the `outputPanel` only being shown while the player is not matched.

A second `outputPanel` is only shown while the player is matched. This panel contains the user interface of the game, which consists of a <a4j:repeat> component (displaying the messages), a text box and a submit button. The code for these user interface elements is the same as for any SEAM application and is therefore not discussed in detail.

Two additional components must be added to any page which uses the multi-player framework:

```
<a4j:poll
  action="#{gwapGameTestPlayer.poll(gwapGameTestSharedGame.currentRound)}"
  interval="2000" />
```

The `poll` component handles player timeouts and redirects of the `Player` base class and allows the AI to operate independently of the player.

The `push` component is used for redirects which are issued by the `Player` subclass of a game. This is achieved by the `action` attribute:

```
<a4j:push action="#{gwapGameTestPlayer.notified}"
  reRender="chatpanel
  #{gwapGameTestPlayer.isNotified('justMatched')?',gameform':''}"
  eventProducer="#{gwapGameTestPlayer.addListener}" interval="500" />
```

More importantly, the `push` component allows notifications to be sent to a player's interface, with little latency and overhead. After the `listener` has been registered with the `Player` object (through the `eventProducer` attribute), the `signal` functions can send notifications to the player. When such a notification has been received, the `push` component updates the user interface elements listed in the `reRender` attribute. Thus, it is possible to selectively update parts of the user interface by using the `isNotified` function of the `Player` object. In the example, the `chatpanel` containing the messages is always updated. Additionally, the entire `gameform` is updated if the `justMatched` notification has been received. This notification is sent by the `PlayerMatcher` after a waiting player has been matched with a partner. Thus, when `justMatched` has been received and the entire user interface is updated, the panel displaying the waiting notification is replaced by the panel with the user interface.

Further Steps. This concludes the creation of a minimalistic game. Many additional features need to be implemented for a real game, such as handling of scores, processing moves in turn-based games or customizing game types. However, these topics are beyond the scope of this introduction. The more advanced features of the framework are best explored by reading the source code of the framework and the KARIDO implementation.

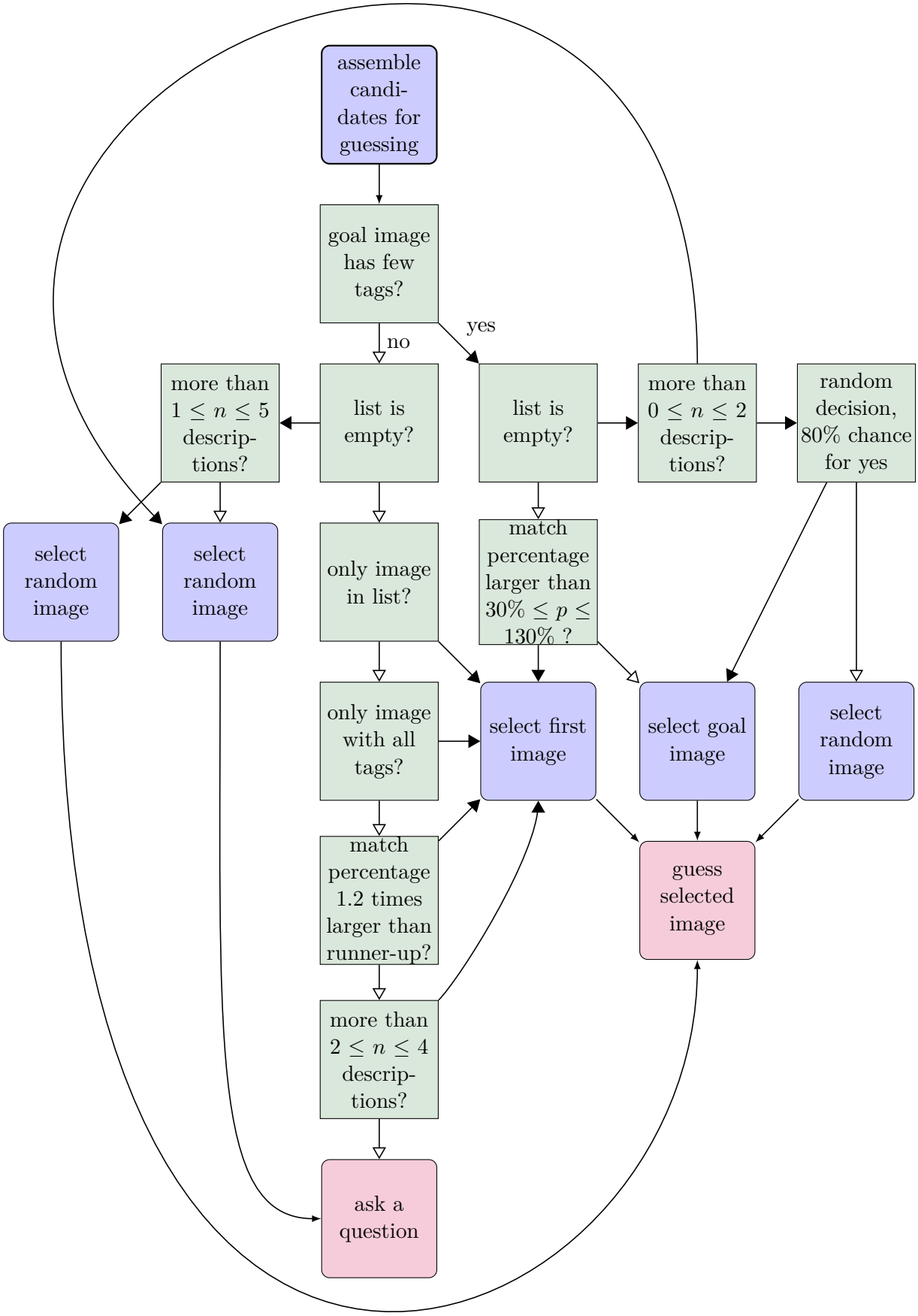


Figure 11: Flow chart illustrating the behavior of the simulated Guesser in Karido. Blue and red nodes indicate actions, with red nodes signifying ends of the flow. Green nodes represent decisions. Lines from decision nodes lead to the possible outcomes with filled arrowheads representing "yes".

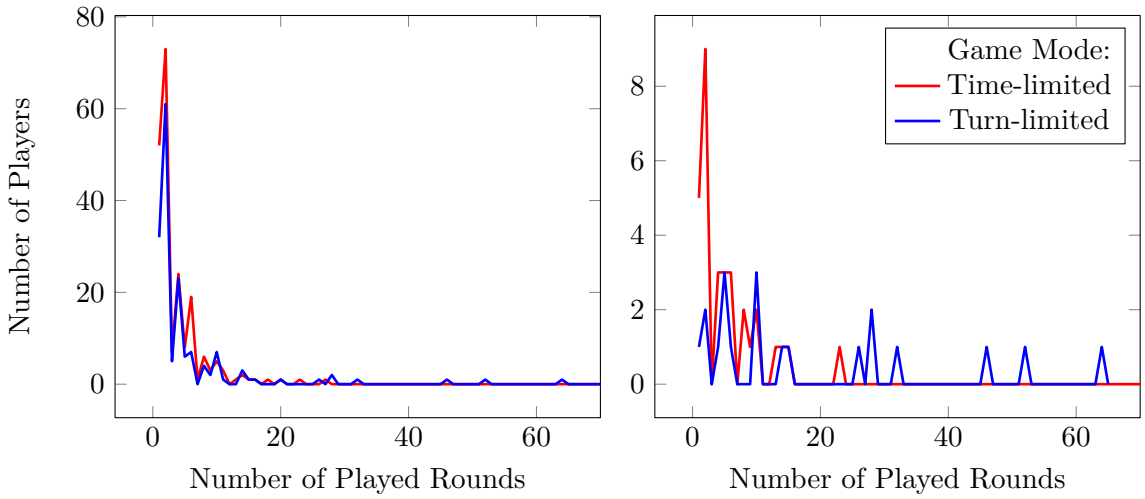
5. Conclusion and Outlook

The following Section concludes this thesis. It presents the results of a short empirical evaluation of KARIDO. Additionally, this Section describes potential future enhancements to the game which were collected during the evaluation but exceed the scope and temporal constraints of this thesis.

5.1. Evaluation

To assess whether the goals of KARIDO described in Section 1.3 were met, an empirical evaluation of the game has been performed. As mentioned in Section 4.1, KARIDO is based on the ARTIGO platform. The current version 4.0 of ARTIGO was released to the public on April 29th, 2011, after a previous testing phase. KARIDO was released alongside ARTIGO. No public announcement of the release was made until May 3rd, 2011. Thus, the players of the early public phase are primarily regular players of ARTIGO, returning to the website. Therefore, the number of the participants of the evaluation is low. However, because of the timing restrictions imposed on this thesis, extending the evaluation phase was not feasible. A snapshot of the database of ARTIGO and KARIDO was taken on May 3rd, 2011 and used to extract the following results. The results should be considered preliminary because of the small number of participating players.

5.1.1. Results



(a) All players (including anonymous players)

(b) Registered players only

Figure 12: Histogram of the number of played rounds in evaluation of KARIDO.

The evaluation phase of KARIDO lasted approximately three weeks, from April 13th, 2011 through May 3rd, 2011. During this period, 301 players completed at least one round of the KARIDO. The participants played a total of 977 game sessions, consisting of 1783

rounds. In the same timespan, 5766 rounds of Artigo were played. The higher popularity of Artigo is likely caused by the profile of the players participating in the evaluation. As the participants consist of regular players of Artigo, a preference for this game has to be expected. However, this discrepancy must be kept in mind when comparing the results of the two games.

Figure 12a shows a histogram of the number of played rounds for both modes of Karido. Strong peaks exist for even numbers of rounds. These peaks are caused by the fact that each game session lasts two rounds. The pronounced peaks indicate that the players tend to finish a game session (as opposed to leaving the game prematurely). The histogram also shows that the majority of players completed only one game session. The average number of completed rounds for the game mode imposing a time limit is 4.48 (considering only players who completed at least one round). The turn-limited game mode proves to be slightly more popular, with some players having completed over 40 rounds of this mode. However, these players are few and thus the average number of played rounds in this mode is 5.17.

The low average number of played rounds indicates that players do not enjoy playing the game. However, an alternative explanation can be found in the way players are identified by the game. Players may register an account, which allows them to be identified upon returning to the Website. Thus, the number of played rounds can be counted accurately for registered players. However, players may also choose to play the game without registering. In this case, only the number of played rounds during one visit to the Website can be counted. This number must necessarily be less than or equal to the total number of played rounds of a player. Figure 12b shows a histogram of the number of played rounds of all registered players. The number of samples is too small to draw reliable conclusions, however, the distribution of played rounds of registered players appears to be more even than the distribution of played rounds of all players.

Another issue potentially contributes to the low average number of played rounds: As a result of the small number of participants, only three game sessions were played by a pair of human players. All remaining sessions were played by a human and a simulated player. This situation is expected in the immediate time after the release of a game. Additionally, the data collected by the game is independent of whether a player is matched with a human or a bot. However, the simulated player cannot substitute a human adequately in all situations of the game. Therefore, game rounds involving to human players can be expected to be more entertaining and therefore lead to higher user participation.

Subjective Player Satisfaction To measure the subjective enjoyment players gain from Karido, a rating mechanism was added to the user interface. After each completed game session, players are invited to “rate this game session”. To submit their rating, players can select from a scale of five stars. This scheme is used on many Websites to express ratings from dislike (one star) to approval (five stars). A total of 568 ratings were created. As described above, 974 game sessions were played by a human and a simulated player, implying at most one rating per game session. Thus, a rating was submitted for 58% of all sessions.

Figure 13 contains a histogram of the submitted ratings for both modes of Karido. In contrast to the low average number of played rounds (indicating low acceptance of the game),

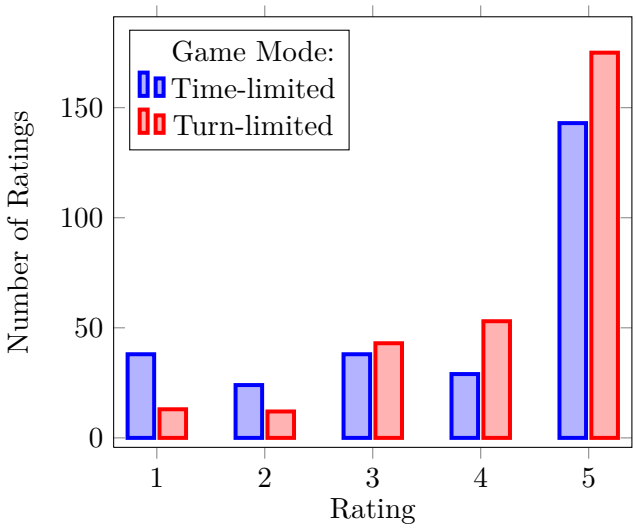


Figure 13: Histogram of the game session ratings submitted during the evaluation.

the submitted ratings are positive. The histograms of both game modes show a strong peak for the maximum rating of 5. As before, the turn-limited mode is slightly more popular, with an average rating of 4.2 (as opposed to 3.8 for the time-limited mode). One possible explanation for the high ratings lies in the way they are collected: As described before, the rating interface is displayed after each completed game session. Therefore, players who do not complete a session do not get a chance to vote. Additionally, players who dislike the game and stop playing only get to vote once, whereas players who like the game and play many sessions can vote more often.

To eliminate the influence of multiple votes, the collected ratings can be averaged in a grouped fashion. First, the average rating for each player is calculated. This reduces the number of ratings to 154, indicating that some players did indeed vote multiple times. Then, the average of these ratings can be calculated, attributing an equal weight to the votes of each player. This results in an average score of 3.7 for the time-limited mode and 4.4 for the turn-limited mode. Thus, the influence of multiple votes is negligible. As described before, players visiting the game's Website multiple times cannot always be identified accurately. Therefore, the votes of some players could still be given too much weight. However, considering the reduced number of ratings after grouping the data, this effect also appears to be negligible.

A comparison between Karido and Artigo is not possible, because no game session ratings were collected for the latter.

Collected Tags Over the course of the evaluation, 8331 labels were created. This number includes any labels that were assigned multiple times. These labels constitute a set of 3826 unique tags. On average, each player of Karido submitted 4.8 tags and 2.1 unique tags per round. During the same evaluation period, Artigo collected 62914 submissions from a set of 9323 unique tags. This corresponds to an average of 10.9 tags and 1.6 unique tags per round. The "tagging rate" (meaning the average number of non-unique tags per round)

is slightly more than twice as large than the rate for KARIDO.

The difference in the tagging rates can be explained by the gameplay of KARIDO, which results in each player only creating descriptions every second round. While questions posed by the Guesser can also create tags, the number of tags created in this fashion is low for two reasons. Firstly, asking questions is only necessary in the case of imprecise descriptions, whereas describing tags must necessarily be given. Secondly, only questions answered with "yes" are converted into tags. Therefore, the influence of answered questions on the tagging rate is low.

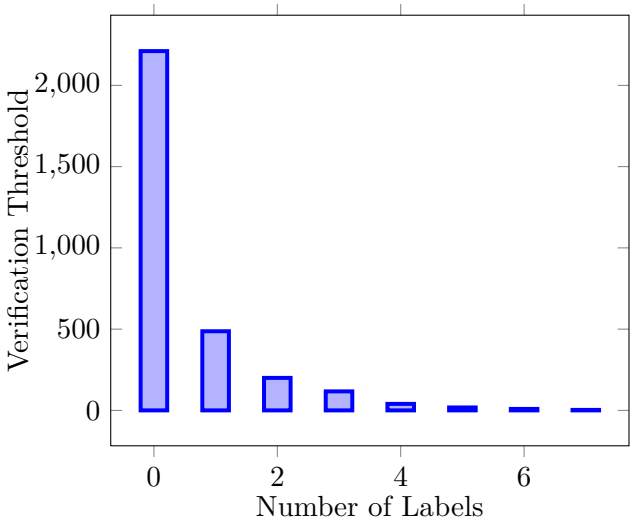


Figure 14: Histogram of the validation scores of the tags collected during the evaluation. Each bar represents alls tags whose rating lie between the label value and the label value plus one (thus, the first bar represents all tags which have a score s of $0.0 \leq s \leq 1.0$).

As described in Section 3.2.1, KARIDO uses a verification scheme to collect only correct tags. This scheme uses a real-valued score to measure the validity of a given tag. Overall, 871 unique, *verified* tags were collected. In the remainder of this paragraph, a tag is considered verified when its verification score is greater than or equal to 1.0. Figure 14 contains a histogram of the scores assigned to all labels. The first bar represents unverified tags with a verification score less than 1.0. This set is by far the largest group of tags and includes two types of unverified labels: Firstly, tags that were not used often enough to raise their verification scores. Secondly, tags that were used several times, but did not lead to successful selection of the goal image. Currently, the data model of KARIDO does not allow to distinguish these two types of tags.

As before, caused by the small number of participants in the evaluation, the collected data is insufficient to draw meaningful conclusions. After the evaluation period, only 12 images were tagged with at least five verified labels. The validity of these tags has not been evaluated, because of the very small sample size.

Play Time and Played Resources On average, the game sessions of KARIDO during the evaluation lasted 4 minutes 5 seconds and 4.8 images were played (4 played images correspond to 3 correct guesses). The average session length of time-limited games is 2 minutes 48 seconds. In this time, an average of 3.5 images were played. The timespan closely corresponds to the 3 minute time-limit enforced by the game. The missing time in the average session length can be attributed to players who prematurely left a game session or were disconnected.

The average session length for the turn-limited game mode is 5 minutes 31 seconds, with an average number of 5.9 played images. This could indicate one of two things: Firstly, the limit of 30 moves could be too large in comparison to the time limit, leading to a simpler game and thus more successful players. Secondly, the players could indeed use the additional time available in the turn-based mode to create more descriptive tags and thus be more successful. The second explanation could be supported analyzing the tags collected by both games, if enough tags had been collected.

The average session length of ARTIGO is 4 minutes 36 seconds. This time also closely corresponds to the enforced time-limit of 5 minutes.

5.2. Future Work

KARIDO is completed as described in Section 3 and working without issues as far as tested. The preliminary evaluation indicates that the game is fun play and collects reasonable data. However, several possible tasks remain to be completed in the future.

5.2.1. Large Scale Evaluation

KARIDO is designed to collect comprehensive sets of tags. An empirical evaluation of this design goal should be performed. However, KARIDO's main mechanism of input-similarity only comes into full effect with reasonably sized collections of images (as there must be a sufficient number of similar images). Furthermore, the game strives to balance the number of tags for each image, processing images with few tags first. Thus, a large number of generic tags have to be applied to all images before more specific labels are collected. A proper evaluation of the game therefore requires a large number of participants or a long duration, preferably both.

Because of the time constraints of this thesis and the coordinated release of both ARTIGO and KARIDO, no large scale evaluation of the created game could be performed. However, all data created since the game's release in April 2011 is collected and at the time of writing, user numbers are still increasing. Thus, a considerable amount of data will be collected in the following months, allowing to evaluate KARIDO in a statistically significant way and comparing its results to those generated by ARTIGO.

5.2.2. Improved Data Verification

As described before, KARIDO uses a scoring system to assess the validity of any tags entered by the players. Tags with a score exceeding a given threshold can be considered reliable.

However, there is currently no way to distinguish between tags which have a low score because they are wrong and tags which have not yet been validated. The game should not continue to validate tags which seldom or never lead to correct guessing, in order to avoid annoying the Guessers. Similarly, tags which have been verified to be correct should also be verified no longer, to ensure that new tags can be verified quickly.

The data model of KARIDO could be enhanced to keep track of the number of times a given tag has been replayed. The verification could then be stopped after a tag has been replayed a certain number of times. However, this would enable malicious players to remove tags from the verification pool, by deliberately entering wrong tags. Alternative schemes could use median values of the number of times all tags have been replayed to balance the verification process.

Designing and implementing a verification scheduling system for large datasets exceeds the scope of this thesis. However, such a system could be created in the future to improve the data verification process.

5.2.3. Compiling Compound Tags

During the creation and evaluation of KARIDO, a new issue for image labeling games emerged. While KARIDO ensures that its players provide new labels, it does not impose restrictions on the form in which these labels are applied. For example, consider a set of images played in a round of the game. Assume these images were selected because they were all tagged “brown”. The images contain brown elements, for example a brown dog, a brown car or simply brown as a dominating color. Both players are likely to recognize the common theme of the images. Therefore, describing the common element of the images does not increase the chance of the Guesser selecting the right image. As a result, the Describer is much more likely to send “dog” than “brown dog” to describe an image depicting a brown dog. In this case, the tag “dog” is implicitly linked to the existing label “brown”. However, this link is not reflected in the stored labels. This means some information introduced by the player is lost.

This loss of information can be a problem, for example in the case of image retrieval. A user wanting to find an image depicting a brown dog will likely search for the label “brown” and ”dog”. Assume there are two images in the database, one depicting a brown dog (tagged “brown”, “dog”), the other depicting a brown car and a black dog (tagged “black”, “brown”, “car”, “dog”). The first image is much more relevant to the query. However, using the labels collected by KARIDO, both images will be considered equally relevant.

It is therefore desirable to collect not only tags, but also semantic associations between them. As described above, such associations are sometimes implicitly defined by the tags based on which the images of a round were selected. However, this relationship is not always clear-cut. For example, an image containing a red car in front of a brown background could be selected for a set of images containing the tag “brown”. If a player described this image, she would likely use a tag such as “car”. This tag should not be associated with the label “brown”. Whether or not a tag entered by the Describer is related to the labels used to select the images cannot be determined without considering the content of the images – which is the reason for creating Games With A Purpose in the first place.

As collecting associations between tags in KARIDO is not possible, the game should be modified to either incite its players to enter compound tags or to explicitly collect associated tags. The rest of this Section outlines a game designed to collect associations between tags.

The basic layout of KARIDO remains unmodified (see Section 3.1). The players assume the roles of Describer and Guesser and are shown a grid of images. The Guesser's task is to select an image that is described by the Describer. The images in the grid are selected by similarity. In contrast to the basic design of KARIDO, the Describer cannot enter tags directly. Instead, the Describer is given a list of all labels assigned to the current goal image. To describe the image, the Describer must select a pair of labels by clicking them. This pair of tags is sent to the Guesser and an association between the tags is stored.

While this approach is relatively simple, several details have to be considered when implementing this new game:

- A suitable data structure for storing the associations has to be created. It must be decided whether the order of the association should be maintained (for example, "red car" and "car red" could be identical or different associations).
- After an association has been confirmed a number of times, this association should also be displayed to the Describer to allow the creation of longer chains of labels. This could be achieved by creating a new label containing both previous tags. When performing this combination of tags, the order of the tags must be decided. Additionally, the stored association should not be discarded. Longer chains of associations can result in complex association graphs, including circular associations. These cases must be handled when creating linear sequences of tags.
- In many cases, an association between two tags will not yield a grammatically correct label, especially in languages other than English. For example, "wood" and "house" might be combined to "wooden house". Automatically adapting the individual labels is difficult. Alternatively, the Describer could be allowed to edit the selected labels. However, it should be ensured that the stem of the labels is not modified. Enforcing this restriction while retaining the flexibility to combine arbitrary labels correctly is difficult. Alternatively, the grammatical errors in combined labels could be accepted. However, especially for longer chains of labels this might bother the players of the game.
- It might be desirable to visualize all associations collected for an image. As the collected associations form a graph, traditional graph embedding techniques such as spring embedding could be used. Whether a node-link visualization is successful for nodes containing individual tags would have to be evaluated.

Finding appropriate solutions to these tasks exceeds the scope of this thesis. The game proposed above has therefore not been implemented and evaluated. Extending KARIDO to collect associations of labels is left as future work.

5.3. Summary And Contribution

The goal of this diploma thesis is to create a novel image labeling game, collecting more comprehensive tag sets than previous implementations. This thesis describes the progress

of the creation of this game, including previous work on which KARIDO is based, the design process and the implementation. A short evaluation has shown promising results, especially for the time-limited mode of the game. However, more assessments are necessary to conclusively prove the effectiveness of the game.

The scientific contribution of this thesis is a new design of an image labeling game, which can potentially collect more comprehensive tags sets and has been proven to be fun to play in a empirical evaluation. The game is publicly accessible at the time of writing and collects valuable data. This data could be used in the future to compare KARIDO and ARTIGO as well as to describe the labeled images. Additionally, a framework for creating multi-player games in SEAM was developed, tested and demonstrated as part of this thesis.

A. Appendix

A.1. Source Code of Example Game

A.1.1. Ai.java

```
package gwap.game.test;

import org.jboss.seam.ScopeType;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Scope;

@Name("gwapGameTestAi")
@Scope(ScopeType.CONVERSATION)
public class Ai extends Player {
  private static final long serialVersionUID = 1L;

  @Override
  public boolean isAi() {
    return true;
  }

  @Override
  public boolean isTimedOut() {
    return false;
  }

  @Override
  public synchronized String poll(int round) {
    setMessage("This is the AI!");
    sendMessage();
    return "";
  }
}
```

A.1.2. Player.java

```
package gwap.game.test;

import org.jboss.seam.ScopeType;
import org.jboss.seam.annotations.In;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Scope;

@Scope(ScopeType.CONVERSATION)
@Name("gwapGameTestPlayer")
public class Player extends gwap.game.Player<Player> {

  private static final long serialVersionUID = 1L;

  @In
  private SharedGame gwapGameTestSharedGame;
```

```
  private String message;

  public Player() {
    setAllowAi(true);
  }

  public String getMessage() {
    return message;
  }

  public void setMessage(String message) {
    this.message = message;
  }

  public void sendMessage() {
    gwapGameTestSharedGame.sendMessage(message);
    message = "";
    signalPartner();
  }

  @Override
  public synchronized String notified() {
    if (isNotified("endGame")) {
      return "home";
    }

    return null;
  }
}
```

A.1.3. PlayerMatcher.java

```
package gwap.game.test;

import gwap.tools.Pair;

import org.jboss.seam.ScopeType;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Out;
import org.jboss.seam.annotations.Scope;

@Scope(ScopeType.APPLICATION)
@Name("gwapGameTestPlayerMatcher")
public class PlayerMatcher extends gwap.game.PlayerMatcher<SharedGame, Player>
    {

  private static final long serialVersionUID = 1L;

  @Out(required = true)
  private SharedGame gwapGameTestSharedGame;

  @Out(required = true)
  private Player gwapGameTestPlayer;
```

```
  public PlayerMatcher() {
    super("gwapGameTest");
  }

  public void enqueue() {
    Pair<SharedGame, Player> p = match();
    gwapGameTestSharedGame = p.a;
    gwapGameTestPlayer = p.b;
  }
}
```

A.1.4. SharedGame.java

```
package gwap.game.test;

import java.util.ArrayList;
import java.util.List;

import org.jboss.seam.ScopeType;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Scope;

@Name("gwapGameTestSharedGame")
@Scope(ScopeType.CONVERSATION)
public class SharedGame extends gwap.game.SharedGame<Player> {

  List<String> messages = new ArrayList<String>();

  public SharedGame() {
    messages = new ArrayList<String>();
  }

  @Override
  public boolean isCompatible(gwap.game.SharedGame game) {
    return (game instanceof SharedGame);
  }

  public List<String> getMessages() {
    return messages;
  }

  public void sendMessage(String message) {
    messages.add(message);
  }

  @Override
  public void startNewRound() {
  }
}
```

A.1.5. testGame.xhtml

```
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:s="http://jboss.com/products/seam/taglib"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:rich="http://richfaces.org/rich"
  xmlns:a4j="http://richfaces.org/a4j"
  template="layout/template.xhtml">


  <ui:define name="body">
    <a4j:form id="gameform">

      <a4j:outputPanel rendered="#{!gwapGameTestPlayer.matched}">
        <ui:include src="widget/waitPlayer.xhtml" />
      </a4j:outputPanel>

      <a4j:outputPanel rendered="#{gwapGameTestPlayer.matched}">

        <a4j:queue size="5" sizeExceededBehavior="dropNext" />


        <a4j:outputPanel id="chatpanel">
          <a4j:repeat value="#{gwapGameTestSharedGame.messages}" var="_msg">
            <h:outputText value="#{_msg}" />
            <br />
          </a4j:repeat>


          <h:inputText id="txtbox" value="#{gwapGameTestPlayer.message}" />
          <a4j:commandButton id="chatButton"
            action="#{gwapGameTestPlayer.sendMessage}" reRender="chatpanel"
            value="Send" focus="txtbox">
            <s:defaultAction />
          </a4j:commandButton>
        </a4j:outputPanel>
      </a4j:outputPanel>


      <a4j:push action="#{gwapGameTestPlayer.notified}"
        reRender="chatpanel
      #{gwapGameTestPlayer.isNotified('justMatched')?',gameform':''}"
        eventProducer="#{gwapGameTestPlayer.addListener}" interval="500" />

      <a4j:poll
        action="#{gwapGameTestPlayer.poll(gwapGameTestSharedGame.currentRound)
            }"
        interval="2000" />
    </a4j:form>
  </ui:define>
</ui:composition>
```

References

[1] P. N. Bennett, D. M. Chickering, and A. Mityagin. Picture this: preferences for image search. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 25–26, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-672-4. doi: 10.1145/1600150.1600157. URL http://dx.doi.org/10.1145/1600150.1600157. 18

[2] M. Borella. Source models of network game traffic. *Computer Communications*, 23 (4):403–410, Feb. 2000. ISSN 01403664. doi: 10.1016/S0140-3664(99)00197-8. URL http://dx.doi.org/10.1016/S0140-3664(99)00197-8. 46

[3] T. Chen, M. M. Cheng, P. Tan, A. Shamir, and S. M. Hu. Sketch2Photo: internet image montage. *ACM Trans. Graph.*, 28(5):1–10, December 2009. ISSN 0730-0301. doi: 10.1145/1618452.1618470. URL http://dx.doi.org/10.1145/1618452.1618470. 4

[4] R. Clarke, J. Burken, J. Bosworth, and J. Bauer. X-29 flight control system: lessons learned. *International Journal of Control*, 59(1):199–219, 1994. ISSN 0020-7179. 1

[5] S. Cooper, A. Treuille, J. Barbero, A. L. Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, and Z. Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-937-4. doi: 10.1145/1822348.1822354. URL http://dx.doi.org/10.1145/1822348.1822354. 20, 21

[6] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the World-Wide Web. *Commun. ACM*, 54:86–96, Apr. 2011. ISSN 0001-0782. doi: 10.1145/1924421.1924442. URL http://dx.doi.org/10.1145/1924421.1924442. 1

[7] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. PhotoSketch: a sketch based image query and compositing system. In *SIGGRAPH 2009: Talks*, SIGGRAPH '09, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-834-6. doi: 10.1145/1597990.1598050. URL http://dx.doi.org/10.1145/1597990.1598050. 4

[8] D. Gavilan, S. Saito, and M. Nakajima. Sketch-to-collage. In *ACM SIGGRAPH 2007 posters*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. doi: 10.1145/1280720.1280759. URL http://dx.doi.org/10.1145/1280720.1280759. 4

[9] C. Gentry, Z. Ramzan, and S. Stubblebine. Secure distributed human computation. In *Proceedings of the 6th ACM conference on Electronic commerce*, EC '05, pages 155–164, New York, NY, USA, 2005. ACM. ISBN 1-59593-049-3. doi: 10.1145/1064009.1064026. URL http://dx.doi.org/10.1145/1064009.1064026. 24, 25

[10] S. Hacker and L. von Ahn. Matchin: eliciting user preferences with an online game. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1207–1216, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518882. URL http://dx.doi.org/10.1145/1518701.1518882. 18

[11] C. J. Ho, T. H. Chang, and J. Y. J. Hsu. PhotoSlap: a multi-player online game for semantic annotation. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1359–1364. AAAI Press, 2007. ISBN 978-1-57735-323-2. URL http://portal.acm.org/citation.cfm?id=1619863. 15

[12] C. J. Ho, T. H. Chang, J. C. Lee, J. Y. jen Hsu, and K. T. Chen. KissKissBan: a competitive human computation game for image annotation. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 11–14, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-672-4. doi: 10.1145/1600150.1600153. URL http://dx.doi.org/10.1145/1600150.1600153. 16, 17

[13] S. Jain and D. Parkes. A Game-Theoretic Analysis of Games with a Purpose. In C. Papadimitriou and S. Zhang, editors, *Internet and Network Economics*, volume 5385 of *Lecture Notes in Computer Science*, chapter 40, pages 342–350. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-92184-4. doi: 10.1007/978-3-540-92185-1_40. URL http://dx.doi.org/10.1007/978-3-540-92185-1_40. 5, 24

[14] E. Law and L. von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1197–1206, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518881. URL http://dx.doi.org/10.1145/1518701.1518881. 21, 29, 30

[15] E. Law, A. Mityagin, and M. Chickering. Intentions: a game for classifying search query intent. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI '09, pages 3805–3810, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4. doi: 10.1145/1520340.1520575. URL http://dx.doi.org/10.1145/1520340.1520575. 23, 24

[16] E. Locke, K. Shaw, L. Saari, and G. Latham. Goal setting and task performance: 1969-1980. *Psychological Bulletin*, 90(1):125–152, 1981. ISSN 0033-2909. 37

[17] H. Ma, R. Chandrasekar, C. Quirk, and A. Gupta. Page hunt: improving search engines using human computation games. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 746–747, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1572108. URL http://dx.doi.org/10.1145/1571941.1572108. 23

[18] T. Malone. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*, pages 162–169. ACM, 1980. ISBN 0897910249. 29

[19] S. Matyas, C. Matyas, C. Schlieder, P. Kiefer, H. Mitarai, and M. Kamata. Designing location-based mobile games with a purpose: collecting geospatial data with CityExplorer. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, pages 244–247, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-393-8. doi: 10.1145/1501750.1501806. URL http://dx.doi.org/10.1145/1501750.1501806. 23

[20] T. Paek, Y.-c. Ju, and C. Meek. People Watcher: A Game for Eliciting Human-Transcribed Data for Automated Directory Assistance. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.6755. 22

[21] K. Siorpaes and M. Hepp. Games with a Purpose for the Semantic Web. *IEEE Intelligent Systems*, 23(3):50–60, May 2008. ISSN 1541-1672. doi: 10.1109/MIS.2008.45. URL http://dx.doi.org/10.1109/MIS.2008.45. 23

[22] D. G. Stork. Open data collection for training intelligent software in the Open Mind Initiative. *IEEE Expert Systems and Their Applications*, 14:16–20, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.1983. 1, 2

[23] R. van Engelen, L. Wolters, and G. Cats. Tomorrow's weather forecast: Automatic code generation for atmospheric modeling. *Computational Science & Engineering, IEEE*, 4 (3):22–31, 2002. ISSN 1070-9924. 1

[24] L. von Ahn. Human Computation. *Data Engineering, International Conference on*, 0:1–2, 2008. doi: 10.1109/ICDE.2008.4497403. URL http://dx.doi.org/10.1109/ICDE.2008.4497403. 1, 2, 3, 8, 9, 10, 11

[25] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8. doi: 10.1145/985692.985733. URL http://dx.doi.org/10.1145/985692.985733. 3, 4, 6, 11, 12, 13, 14, 31, 35

[26] L. von Ahn and L. Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008. ISSN 0001-0782. 9, 35

[27] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, chapter 18, page 646. Springer Berlin / Heidelberg, Berlin, Heidelberg, May 2003. ISBN 978-3-540-14039-9. doi: 10.1007/3-540-39200-9_18. URL http://dx.doi.org/10.1007/3-540-39200-9_18. 2, 3

[28] L. von Ahn, S. Ginosar, M. Kedia, R. Liu, and M. Blum. Improving accessibility of the web with a computer game. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 79–82, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7. doi: 10.1145/1124772.1124785. URL http://dx.doi.org/10.1145/1124772.1124785. 17, 18

[29] L. von Ahn, M. Kedia, and M. Blum. Verbosity: a game for collecting common-sense facts. In *In Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems, volume 1 of Games*, pages 75–78, 2006. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.577. 7, 19, 20, 27, 33

[30] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A Game for Locating Objects in Images. In *Proceedings of the SIGCHI conference on Human Factors in computing*

systems, pages 55–64. ACM, 2006. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.4835&rep=rep1&type=pdf. 14, 15

[31] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895): 1465–1468, September 2008. doi: 10.1126/science.1160379. URL http://dx.doi.org/10.1126/science.1160379. 3

[32] I. Weber, S. Robertson, and M. Vojnović. Rethinking the ESP game. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI '09, pages 3937–3942, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4. doi: 10.1145/1520340.1520597. URL http://dx.doi.org/10.1145/1520340.1520597. 5, 6, 17

[33] J. Yan and S. Y. Yu. Streamlining attacks on CAPTCHAs with a computer game. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 2095–2100, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc. URL http://portal.acm.org/citation.cfm?id=1661445.1661780. 19

[34] S. Yasunobu and S. Miyamoto. Automatic train operation system by predictive fuzzy control. *Industrial Applications of Fuzzy Control*, pages 1–18, 1985. 1

Web References

[art] Artigo. http://www.artigo.org.

[bin] Bing (formerly Live) Image Search. http://www.bing.com/images.

[gooa] Google Image Labeler. http://www.images.google.com/imagelabeler.

[goob] Google Images. http://www.images.google.com.

[Hasa] Hasbro. MEMORY instruction manual. http://www.hasbro.com/common/instruct/Memory_Original_Collector's_Tin_2006.pdf.

[Hasb] Hasbro. TABOO instruction manual. http://www.hasbro.com/common/instruct/Taboo_(1989).pdf.

[pro] Prometheus Bildarchiv. http://www.prometheus-bildarchiv.de.

[sea] Seam. http://www.seamframework.org.

[sky] Skype. http://www.skype.com.

[tin] TinEye. http://www.tineye.com.

[wik] Wikipedia. http://www.wikipedia.org.

[you] YouTube. http://www.youtube.com.