RecTurk: Constraint-based Recommendation based on Human Computation*

Alexander Felfernig  
Institute for Software Technology  
A-8010 Graz, Austria  
aelfern@ist.tugraz.at

Sarah Haas  
Institute for Software Technology  
A-8010 Graz, Austria  
sarah.haas@student.tugraz.at

Gerald Ninaus  
Institute for Software Technology  
A-8010 Graz, Austria  
gninaus@ist.tugraz.at

Michael Schwarz  
Institute for Software Technology  
A-8010 Graz, Austria  
michael.schwarz@student.tugraz.at

Thomas Ulz  
Institute for Software Technology  
A-8010 Graz, Austria  
thomas.ulz@student.tugraz.at

Martin Stettinger  
Institute for Software Technology  
A-8010 Graz, Austria  
mstettinger@ist.tugraz.at

ABSTRACT

The development of constraint-based recommender applications is still challenged by the knowledge acquisition bottleneck. The management of the underlying constraint sets is often accompanied by additional efforts related to the information exchange between knowledge engineers and domain experts. In this paper we introduce the RecTurk research prototype which supports the development of constraint-based recommender applications on the basis of Human Computation concepts. Thus we substitute complex knowledge engineering tasks with simple micro-tasks that can be performed by persons without experiences in constraint-based recommender application development. We introduce the basic concepts currently integrated in RecTurk and report the results of a first user study that evaluated the applicability of RecTurk in three item domains.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous; D.2.2 [Design Tools and Techniques]: User interfaces

General Terms
Human Factors, Algorithms

Keywords
Constraint-based Recommendation, Human Computation

1. INTRODUCTION

In this paper we present approaches that support an easier development of constraint-based recommenders [3, 5]. These systems are beside critiquing-based recommenders [1, 2] a major representative of knowledge-based recommenders [1].

*This work presented in this paper has been conducted in the project PeopleViews funded by the Austrian Research Promotion Agency (843492).

Critiquing-based recommenders focus on a navigation-based paradigm where users are navigating in the item (product) space by articulating critiques, for example, a cheaper camera or a more fancy car and the recommender retrieves new alternatives on the basis of similarity metrics. Constraint-based recommenders focus on a search-based user support where users articulate requirements and the recommender system searches for a solution.

In the following we focus on constraint-based recommenders [3] that exploit constraints (rules) in order to determine recommendations. Such systems are especially useful in scenarios related to high-involvement items where consumers typically invest more time and efforts (e.g., evaluation of prices, quality, and services) to assure they make the best choice. They are prepared to invest additional efforts since the purchase decision is often associated with high investment costs and/or is related to increases in or losses of prestige. The underlying items are typically complex which requires constraint-based representations. Examples of items for which constraint-based recommenders have already been developed, are digital cameras, financial services, and software engineering methods.

The major advantages of constraint-based recommenders are (1) deep knowledge about the item domain which is exploited for taking into account existing domain restrictions (e.g., in the automotive domain certain car bodies must not be combined with certain engines), (2) deep explanations for recommendation results (e.g., the user requirements high return rate and low risk are directly related to the item property longterm investment period), and (3) automated diagnosis and repair of inconsistent requirements (e.g., a user has to change his/her willingness to take risks due to the fact that a short investment period and high return rates are preferred).

One major drawback of constraint-based recommendation technologies are time-intensive knowledge acquisition processes that are needed to identify the item domain knowledge (properties and the corresponding constraints). Efforts related to the communication overhead between item domain experts and knowledge engineers are often summarized (denoted as) knowledge acquisition bottleneck. This phenomenon is one of the major impediments for a more wide-spread application of constraint-based recommenders.
Intuitive knowledge acquisition for constraint-based recommenders can have an enormous impact on user acceptance. Our major goal is to provide engineering techniques that allow persons without technical expertise in the development of constraint-based recommenders to easily develop their own recommenders or contribute to recommenders of interest. In this context we introduce the RECTURK research prototype\(^1\) that helps to easily define and execute constraint-based recommenders on the basis of micro-contributions [8] of a group of users. RECTURK asks simple questions (in the form of micro-tasks [8]) and then (from the given answers) derives a recommendation knowledge base. As a working example in this paper, we will use the domain of digital cameras. However, RECTURK users are free to choose a (reasonable) application domain on their own.

The name RECTURK was inspired by the amazon.com platform amazon mechanical turk which hosts a community that completes so-called Human Computation tasks (see [10]).\(^2\) Typically, such tasks are easy to complete by humans but often infeasible for computers. Examples of such tasks are the quality comparison of pictures or the manual identification of item properties from a web page. Another example of the application of Human Computation is CAPTCHA (Completely Automated Public Turing Test To Tell Computers and Humans Apart – www.captcha.net) which is exploited, for example, to protect websites against bots.

Human Computation style approaches have already been applied in the context of recommender systems. For example, Wikilens [8] is a recommendation environment that provides user communities the possibility to cooperatively develop recommenders in an open environment where users are able to introduce new items, comment on items, and search for items. The basic principles of community-maintained recommenders [8] are taken into account in RECTURK. Another application of Human Computation in the recommendation context is the work of [9] (Matchin). Matchin is based on the idea of eliciting preferences for large datasets by asking users what a "random" person would prefer when having to choose between alternatives. The difference between the work of [8, 9] and RECTURK is that RECTURK-collected and -compiled recommendation knowledge can be exploited for constraint-based recommenders [3] that include functionalities such as deep explanations and intelligent diagnosis and repair for inconsistent requirements.

The remainder of this paper is organized as follows. In Section 2 we sketch the application of Human Computation in constraint-based recommendation. The RECTURK user interface is presented in Section 3. The results of a first empirical study are presented in Section 4. The paper is concluded with Section 5.

### 2. RECTURK APPROACH

On the basis of a working example from the domain of digital cameras we show how RECTURK can be exploited for the development and application of constraint-based recommenders. RECTURK can be used in two basic modes.

#### A. Modeling Mode

Recommender applications (items, properties, and constraints) can be defined and managed in the modeling mode. Users engaged in the modeling mode can contribute to the construction of constraint-based recommenders by simply completing a set of so-called micro-tasks [8]. Micro-tasks currently offered by the RECTURK environment are summarized in Table 1.

<table>
<thead>
<tr>
<th>micro-task</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>define recommender</td>
<td>create a new recommender</td>
</tr>
<tr>
<td>add item</td>
<td>add a new item to recommender</td>
</tr>
<tr>
<td>add item property</td>
<td>add a new item property</td>
</tr>
<tr>
<td>describe item property</td>
<td>characterize selected property(ies)</td>
</tr>
<tr>
<td>rate item</td>
<td>provide a new rating for an item</td>
</tr>
<tr>
<td>rate recommender</td>
<td>evaluate a recommender</td>
</tr>
</tbody>
</table>

#### Table 1: Types of RecTurk micro-tasks.

In RECTURK, items (see, e.g., Table 2) can be added to a recommender and characterized with regard to their basic properties (e.g., whether a digital camera is well-suited for sports photography). Product (item) information is entered directly by users (the add-principle [8]). Each user is allowed to add new items but also to adapt the information of existing ones. Only administrators are allowed to insert, update, and delete item properties. For our example knowledge base we are interested in the following item properties (see Table 3): portrait (support of good portrait pictures), sports (good support of sports photography), macro (good support of macro photography), and price (price segment). Item properties describe the way in which users can evaluate an item and how they can specify their requirements in the recommendation mode. For each item property we have to specify the question posed to the user (see Table 3).

#### Table 2: Example items (CANON EOS cameras).

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
<th>link</th>
<th>user</th>
</tr>
</thead>
<tbody>
<tr>
<td>7D</td>
<td>7D is ...</td>
<td>...</td>
<td>Ed</td>
</tr>
<tr>
<td>Mark III</td>
<td>Mark III is ...</td>
<td>...</td>
<td>Mary</td>
</tr>
</tbody>
</table>

#### Table 3: Example item properties.

<table>
<thead>
<tr>
<th>property</th>
<th>domain</th>
<th>question to user</th>
<th>user</th>
</tr>
</thead>
<tbody>
<tr>
<td>portrait</td>
<td>(yes, no)</td>
<td>portrait shots?</td>
<td>Mary</td>
</tr>
<tr>
<td>sports</td>
<td>(yes, no)</td>
<td>sports shots?</td>
<td>Mary</td>
</tr>
<tr>
<td>macro</td>
<td>(yes, no)</td>
<td>macro shots?</td>
<td>Mary</td>
</tr>
<tr>
<td>price</td>
<td>0–300, ... ≥ 1500</td>
<td>accepted price?</td>
<td>Mary</td>
</tr>
</tbody>
</table>

#### B. Recommendation Mode

Recommender applications (developed in the modeling mode) are available for public use in the recommendation mode. As mentioned, we use the domain of digital cameras as a simple working example – however, in RECTURK users are free to choose their own item domain. For the purposes of our working example we assume the existence of users (see Table 4). These users contribute to the construction of the recommender knowledge base and apply the resulting RECTURK recommender (in our case: a simple digital camera recommender).

**Recommendation Task.** In RECTURK, a recommendation task \( (P, A, F, R) \) consists of an item table \( P \), a set \( A \) of item...
properties (attributes of $P$), a set $F$ of filter constraints that select items from $P$ depending on the requirements in $R$, and a set of customer requirements ($R$). In this context, $P, A, F$ are denoted as recommender knowledge base. The identification of recommendation candidates can be described as a conjunctive query $\sigma_{R,F}[P]$ where $\sigma$ represents the selection operator of a conjunctive query that is executed on the item table $P$ taking into account the selection criteria in $R \cup F$. The basic steps to derive and apply a RecTurk recommender knowledge base will be discussed in an example.

Before we are able to apply a RecTurk recommender, we have to define the relevant properties of this recommender, i.e., the basic elements needed for the definition of a recommendation task. Let us assume that the RecTurk recommender DIGICAMS has already been defined (micro-task define recommender) and that this task has been performed by the user Mary. Thereafter, users start entering items that should be integrated in DIGICAMS from their point of view. In our working example the users Joe, Ed, and Mary add the items (Canon cameras) 300D, 7D, and Mark III (related micro-task: add item – see Table 2). After having created the RecTurk recommender DIGICAMS, Mary also added a set of item properties (see Table 3). An example of an item property is portrait which entails the information whether a camera provides a good support for portrait photography – the corresponding micro-task is add item property. Since only creators of recommenders are currently allowed to define item properties, Mary is the only user who is allowed to do that in the DIGICAMS recommender (see Table 3 for an overview of item properties used in our working example). After item properties have been defined, other users are allowed to characterize items with regard to the defined item properties, for example, user Peter who evaluates the Canon EOS 7D camera, concludes that the camera provides a good macro support (macro = yes). For our example we assume that other users are specifying item properties as well (micro-task: describe item property). An overview of these property specifications is given in Table 4.

**User-specific Filter Constraints.** Each user is allowed to rate each RecTurk item; a user-specific rating provided for an item is denoted as user-specific filter constraint. Such constraints reflect item evaluations of a specific user and thus form the basis for defining the relationship between user requirements (instantiations of item properties) and items from the viewpoint of a user. For example, user Mary (see Table 4) recommends the camera Mark III to users interested in good portrait and sports photos. Her rating for this camera is 5.0 (on a scale of 1.0, 2.0, ..., 5.0); she specifies the price category with \( \geq 1500 \), and does not provide an evaluation for the item property macro since she has no further technical information. Users are also allowed to adapt and delete their own user-specific filter constraints. Note that when defining a user-specific filter constraint, there is no need that a user specifies values for all item properties. For example, Mary did not specify a value for the item property macro of the EOS Mark III. It is also allowed that one user specifies more than one value per item property. For example, the 7D can be recommended to users who accept a price of 600-1500 or a price \( \geq 1500 \). On a more formal level, the user-specific filter constraint introduced by Mary would look like as follows (noreq indicates that no value was specified for the property by the recommender user): if (portrait = yes or noreq) and (sports = yes or noreq) and (price \( \geq 1500 \) or noreq) then include (Mark III).

**Recommendation-relevant Filter Constraints.** On the basis of user-specific filter constraints (see, e.g., Table 4), RecTurk applies the following strategy for aggregating these filter constraints to recommendation-relevant filter constraints. For each item, the user-specific filter constraints are aggregated per item property. An example of such an aggregation is shown in Table 5 (aggregation of the user-specific filter constraints related to the item Mark III). On a formal level, the recommendation-relevant filter constraint for the item Mark III looks like as follows (noreq indicates that no value was specified for the item property by the recommender user): if (portrait = yes or noreq) and (sports = yes or noreq) and (price between 600-1500 or price \( \geq 1500 \) or noreq) then include (Mark III).

The complete set of recommendation-relevant filter constraints is shown in Table 6. These constraints are activated in the RecTurk recommendation mode. In RecTurk, filter constraints can automatically be derived from micro-contributions. This way the system is able to automatically derive constraints which represent the knowledge base of a constraint-based recommender. As an alternative to time-consuming communication processes between item domain experts and knowledge engineers, RecTurk provides a reasonable alternative which has the potential to reduce overheads related to the knowledge acquisition bottleneck and – due to a simple way of entering recommendation knowledge – makes constraint-based recommendation technologies accessible for a broader user group.

For each combination of name \( \times \) property value (see Table 4), RecTurk determines a support value (see Formula 1). For example, support(300D, portrait=yes)=1.0 means that the users completely agree on the fact that the Canon EOS 300D camera is a good recommendation when a user wants to take portrait pictures. The complete set of support values for item property values derived from the user-specific filter constraints (Table 4) is included in Table 7. In Formula 1, co-occurrence(name, value) indicates how often the combination of item name and property value can be found. For example, co-occurrence(300D, portrait=yes) = 3.

<table>
<thead>
<tr>
<th>name</th>
<th>rating</th>
<th>portrait</th>
<th>value</th>
<th>sport</th>
<th>price</th>
<th>co-occurrence</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>300D</td>
<td>3.0</td>
<td>yes</td>
<td>no</td>
<td>300D</td>
<td>0-300, 300-600</td>
<td>0-300</td>
</tr>
<tr>
<td>Ed</td>
<td>7D</td>
<td>4.0</td>
<td>yes</td>
<td>yes</td>
<td>600-1500, ( \geq 1500 )</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>Peter</td>
<td>7D</td>
<td>4.0</td>
<td>yes</td>
<td>yes</td>
<td>600-1500, ( \geq 1500 )</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>Ann</td>
<td>7D</td>
<td>4.0</td>
<td>yes</td>
<td>yes</td>
<td>600-1500, ( \geq 1500 )</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>Ed</td>
<td>Mark III</td>
<td>5.0</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>600-1500, ( \geq 1500 )</td>
<td>5.0</td>
</tr>
<tr>
<td>Alex</td>
<td>300D</td>
<td>3.0</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>300-600, 600-1500</td>
<td>3.0</td>
</tr>
<tr>
<td>Leo</td>
<td>300D</td>
<td>3.0</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>0-300, 300-600</td>
<td>3.0</td>
</tr>
<tr>
<td>Mary</td>
<td>Mark III</td>
<td>5.0</td>
<td>yes</td>
<td>yes</td>
<td>( \geq 1500 )</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>Alex</td>
<td>Mark III</td>
<td>5.0</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>600-1500, ( \geq 1500 )</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 4: Example: user-specific filter constraints (\( r = \text{rating}, p = \text{portrait}, s = \text{sports}, m = \text{macro} \)).

---

\(^3\)In the current RecTurk version it is not possible to define the degree of support, however, this is already planned for the next system version.
Table 5: Example: Aggregation of user-specific filter constraints related to item Mark III. The result is a recommendation-relevant filter constraint for the Mark III.

<table>
<thead>
<tr>
<th>user</th>
<th>name</th>
<th>rating (r)</th>
<th>portrait</th>
<th>sports</th>
<th>macro</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ed</td>
<td>Mark III</td>
<td>5.0</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>≥1500</td>
</tr>
<tr>
<td>Mary</td>
<td>Mark III</td>
<td>5.0</td>
<td>yes</td>
<td>yes</td>
<td>-</td>
<td>≥1500</td>
</tr>
<tr>
<td>Alex</td>
<td>Mark III</td>
<td>5.0</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>≥1500</td>
</tr>
<tr>
<td>John</td>
<td>Mark III</td>
<td>5.0</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>600–1500, ≥1500</td>
</tr>
<tr>
<td>Aggregate</td>
<td>Mark III</td>
<td>5.0 (rating)</td>
<td>yes, no</td>
<td>yes, no</td>
<td>no</td>
<td>600–1500, ≥1500</td>
</tr>
</tbody>
</table>

Table 6: Complete set of recommendation-relevant filter constraints derived from the user-specific filter constraints depicted in Table 4.

<table>
<thead>
<tr>
<th>name</th>
<th>rating (r)</th>
<th>portrait</th>
<th>sports</th>
<th>macro</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>300D</td>
<td>3.0</td>
<td>yes</td>
<td>no</td>
<td>no, yes</td>
<td>0–300, 300–600</td>
</tr>
<tr>
<td>7D</td>
<td>4.0</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>600–1500, ≥1500</td>
</tr>
<tr>
<td>Mark III</td>
<td>5.0</td>
<td>yes</td>
<td>no, yes</td>
<td>no</td>
<td>600–1500, ≥1500</td>
</tr>
</tbody>
</table>

Table 7: Support values for item property values derived from user-specific filter constraints (Table 4).

<table>
<thead>
<tr>
<th>name</th>
<th>support</th>
<th>portrait</th>
<th>sports</th>
<th>macro</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>300D</td>
<td>yes</td>
<td>1.0</td>
<td>1.0</td>
<td>no</td>
<td>.66</td>
</tr>
<tr>
<td>7D</td>
<td>yes</td>
<td>1.0</td>
<td>1.0</td>
<td>yes</td>
<td>1.0</td>
</tr>
<tr>
<td>Mark III</td>
<td>yes</td>
<td>1.0</td>
<td>no</td>
<td>.75</td>
<td>75</td>
</tr>
</tbody>
</table>

Table 8: Calculation of the item utilities (Formula 3) of 300D and Mark III for the user requirements [portrait=yes, sports=no]. The EOS 7D is not included since there is no support for the individual user requirement sports=no.

<table>
<thead>
<tr>
<th>name</th>
<th>support</th>
<th>portrait</th>
<th>sports</th>
<th>total</th>
<th>rating</th>
<th>utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>300D</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
<td>3.0</td>
<td>6.0</td>
<td>8.75</td>
</tr>
<tr>
<td>Mark III</td>
<td>1.0</td>
<td>0.75</td>
<td>1.75</td>
<td>5.0</td>
<td>8.75</td>
<td></td>
</tr>
</tbody>
</table>

3. RECTURK USER INTERFACE

In the following paragraphs we provide an overview of the basic functionalities provided by the RecTurk user interface. RecTurk can be operated in two different modes (see Figure 1).\(^5\) First, a recommender is developed (modeling mode) by exploiting the micro-contributions of users to derive a set of recommendation-relevant filter constraints. Users can provide micro-contributions in different ways, for example, by entering new items into the system or by evaluating items that have already been entered by other users. For example, the user selects the item EOS Mark III and specifies values for portrait, sports, macro, and price (a user-specific filter constraint). RecTurk stores the new (user-specific) filter constraint (see, e.g., Table 4) and recalculates the support values of the individual item property values (see Formula 1) and the item-specific average ratings (see Formula 2). In RecTurk, only logged-in users

\(^4\)Redundancy elimination is planned for future versions.\(^5\)A user does not have to specify all properties.
are allowed to provide micro-contributions. Second, RecTurk recommenders (automatically generated from micro-contributions) are open for use, i.e., no login is needed for applying RecTurk recommenders. An example of a RecTurk user interface for the Mobile Games recommender is depicted in Figure 1. This recommender has been developed by the participants of the user study reported in Section 4.

In our working example (recommendation of digital cameras), the user could specify his/her requirements (portrait = yes, sports = no). RecTurk selects relevant items using the table of recommendation-relevant filter constraints (e.g., Table 6) and ranks the results on the basis of Formula 3.

4. EVALUATION

In this section we will first discuss the benefits of Human Computation based approaches compared to centralized ones where one or few knowledge engineers are in charge of developing and maintaining constraint-based recommenders. Thereafter, we report the results of a study that has been conducted at the Graz University of Technology.

Benefits of Human Computation based Approaches. The development of a constraint-based recommender application can cause development efforts in the range of a couple of man months [7]. In contrast, the basic RecTurk applications of our user study have been developed within the scope of two days. The major reason for this increase in efficiency is the provision of easy micro-tasks that are accessible to a larger group of users. From our commercial recommender projects (see, e.g., [7]) we know that scalability is one of the major challenges of constraint-based recommender development (see also [8]). In many scenarios, one or few knowledge engineers are in charge of developing and maintaining knowledge bases. Due to the increasing number of items offered via recommender applications, development and maintenance operations become infeasible for one or few persons. This situation then triggers a demand for keeping recommendation knowledge up-to-date in a more sophisticated fashion. RecTurk is a first step towards more scalable development techniques which will increase the popularity of constraint-based recommenders.

RecTurk usability. This user study has been conducted at the Graz University of Technology. N=161 (15% female, 85% male) interacted with the RecTurk environment and developed three different constraint-based recommender applications (MobileGames, WorldCities, and SkiResorts). The distribution of expertises in the application of recommendation technologies is depicted in Figure 3. Recommendations and item properties have been created before the study started. The micro-tasks (see Figure 1) of the study participants were to enter new items, to describe...
and evaluate items, and to apply one of the three RecTurk recommender applications where the task was to find and select a preferred item. After interacting with the RecTurk environment, users had to fill out a questionnaire based on the system usability scale (SUS) and to answer further questions. The results of the system usability scale (SUS) evaluation of RecTurk are depicted in Figure 2.

**Willingness to contribute.** Besides analyzing usability aspects, we were interested in whether users can imagine to contribute to the development of RecTurk recommenders, and – if yes – to which extent they are willing to support recommender development (in terms of the micro-tasks they had to complete within the scope of the user study). Figure 4 summarizes the user feedback regarding this question of available time for recommender development. The majority of those users who would like to contribute to RecTurk recommender development (69% of all users) would contribute within a time frame of less than 30 minutes per week (56% out of those who want to contribute).

**Further application domains.** Finally, we wanted to know for which item domains study participants have an interest in developing their own RecTurk recommender applications (N = 80). The most prominent item domains were electronic equipment, restaurants, bars, holiday trips, hotels, games, films, computers, and smartphones.

## 5. CONCLUSIONS

Constraint-based recommendation technologies still suffer from the knowledge acquisition bottleneck: knowledge engineers still have to invest huge efforts to be able to guarantee that the developed constraint set is up-to-date. RecTurk technologies help to tackle this challenge by simplifying knowledge engineering tasks. This simplification is achieved by confronting users with simple micro-tasks such as entering an item and evaluating an item with regard to a list of item characteristics. This form of user input is exploited by the RecTurk environment for automatically deriving constraints which are the basis for the application of knowledge-based techniques such as knowledge-based reasoning, intelligent (deep) explanations, and automated diagnosis & repair. Our future work will include the development and comparison of further item ranking mechanisms, the development of additional micro-task patterns (for constraints, conflict resolution, and redundancy elimination [6]), the development of quality assurance and manipulation detection mechanisms, and the design of “games with a purpose” for knowledge base construction in RecTurk.

## 6. ADDITIONAL AUTHORS

Additional authors: Klaus Isak (SelectionArts Ltd., email: k.isak@selectionarts.com), Michael Jeran (Institute for Software Technology, email: mjeran@ist.tugraz.at), and Stefan Reiterer (Institute for Software Technology, email: stefan.reiterer@ist.tugraz.at).

## 7. REFERENCES