# GroupRecoPF: Innovative Group Recommendations in a Distributed Platform

Tom Gross, Christoph Beckmann, Maximilian Schirmer
Faculty of Media
Bauhaus-University Weimar
99423 Weimar, Germany
<firstname.lastname>(at)medien.uni-weimar.de

***Abstract*—Group recommender systems provide groups of users with shared recommendations. They have great potential for easing group decision processes, while at the same time entailing new technological and user-centred challenges. In this paper we present the GroupRecoPF platform providing developers with support for building scalable and user-friendly group recommender systems. It leverages on advanced concepts for group recommendation merging strategy exploration, performance optimisation, session persistence and mobility, as well as open interfaces.**

***Keywords*—Group Recommender Platform; Recommender Algorithms; Distribution; Scalability.**

## I. INTRODUCTION

*Recommender* systems aim to facilitate individual users' decision making by providing them with suggestions of choices to make, even if they lack personal experience of the alternatives [25]. They 'create recommendations tailored to individual users rather than universal recommendations for, well, everyone' [19, p. 15]. They support users in everyday-live situations with the help of sophisticated algorithms. These algorithms rely on a user's preferences as the basis for generating predictions—that is, a user expresses her taste towards the system: explicitly by means of ratings, or implicitly through following recommendations [16, 24].

*Group recommender* systems go beyond the individual user and provide groups of users with a shared recommendation. They try to consider all group members preferences and generate a shared recommendation that fits the whole group. They are rare and mainly support hedonic activities, since these generally occur in pairs or groups (e.g., watching movies, eating out, travelling abroad). For instance, the PolyLens system is an early example of a group recommender systems offering shared movie recommendations to groups [20].

Despite their clear advantages for groups and their performance, group recommender systems entail new recommendation algorithms as well as the their software architecture.

Group recommender *algorithms* need to respect the diverse taste of a group in the generation of recommendations [11, 29]. Group recommender systems typically rely on aggregation algorithms for merging the individual preferences of all group members. From a group process perspective, the algorithms should generate a prediction of the appreciation for the entire group. So, novel approaches for recommender algorithms need to be developed and explored with end-users.

*Software architectures* need to address distribution and scalability [10]. Group recommender systems typically have a distributed architecture and need to address scalability of user- and system-generated requests as well as user- and system-triggered use and re-use of sessions.

In this paper we present the *GroupRecoPF* platform providing developers with support for building user-friendly and scalable group recommender systems. The *GroupRecoPF* platform offers concepts and an implementation supporting developers of group recommender systems for pairs or groups of users who want shared movie recommendations. It leverages on advanced concepts for group recommendation merging strategy exploration, performance optimisation, session persistence and mobility, as well as open interfaces. The platform faces the technological and social challenges of group recommender systems through adaptable and extensible group recommendation merging strategies. It relies on the easy exploration of merging strategies for the development and adaptation of group recommender systems, and it supports parallel processing and sophisticated caching mechanisms for fast group recommendation generation and presentation. The platform offers persistence of ratings, group recommender sessions, and recommendations. This persistence provides history for users and input data for recommendation generation algorithms. Furthermore, the platform provides open interfaces for connecting mobile and desktop group recommender clients. Clients access session-based recommender data and encapsulate the complexity of the recommendation process in an easy-to-use group interaction.

In the next section we give a brief overview of related work. We then introduce the concepts of the *GroupRecoPF* platform for merging strategy exploration, request management, and programming interfaces. We report on the implementation of these concepts in a distributed architecture. Finally, in the conclusions we summarise our contribution and glance at future work.

## II. Background and Related Work

The *GroupRecoPF* platform leverages on previous work concerning traditional single-user recommender systems, group recommender systems, and distributed parallel event notification infrastructures. In this section we glance at the most relevant of this related work.

### A. Single-User Recommender Systems

Single-user recommender systems provide recommendations based on previously specified preferences to individual users. Some frequently mentioned early single-user recommender systems are *Tapestry*, *GroupLens*, and *PHOAKS*.

*Tapestry* [6] was the first system to introduce the *collaborative filtering* approach for supporting users in handling large amounts of messages and documents from mailing lists. It provides a relevance level determined from previous users' reactions to messages they read. Possible reactions are annotations to messages for implicit feedback, or replies to messages for direct feedback. The system processes reactions for later use in filters, and for message delivery.

*GroupLens* [24] follows the concepts of *Tapestry* and generalises its approach to allow diverse external services to connect and transfer their ratings (e.g., netnews). It provides a distributed architecture for better scalability. A matrix-filling approach guarantees for better predictions.

*PHOAKS* (People Helping One Another Know Stuff) [28] helps users finding relevant information by providing recommendations for URLs. It analyses the mentions of URLs in messages of online conversations (e.g., netnews messages) as input data for collaborative filtering, and recommends frequently found URLs.

Although all three approaches use collaborative filtering mechanisms, they are not dedicated to groups of users. As we will describe later, the *GroupRecoPF* platform offers group recommendations using various merging strategies and it employs a single-user recommender system using collaborative filtering in the background.

### B. Group Recommender Systems

Group recommender systems generate and present shared recommendations to pairs and groups of users. Here we sketch interesting approaches for the simple computation of group recommendations, for the merging of single-user recommendations, and for the presentation of group member preferences.

The *MusicFX* system [18] nicely addresses everyday needs of groups of users who want to find a compromise in the music played while they exercise in a fitness centre. It relies on ratings that users have expressed towards a list of music genres, and determines a group recommendation by averaging the genre ratings of all present fitness centre users. Rating values range from –2 for "hate" to +2 for "love". Based on these ratings, the system is able to estimate the users' predictions for music tracks that are associated with these genres.

The *PolyLens* system [20] is a movie group recommender system that generates movie recommendations for groups of users. Based on ratings expressed by users on a movie recommender Website, the system generates recommendations using a collaborative filtering approach. The *PolyLens* system introduces new ways to aggregate the predictions for groups of users to generate group recommendations. The authors propose two different approaches: either a pseudo-user that contains all ratings of the group members, or generating recommendations lists for each user and merging these lists for group recommendations.

The *Travel Decision Forum* recommender system [12] presents a collaborative approach for distributed groups of users who want to find a compromise for a group vacation. The system supports asynchronous communication between spatially separated group members through animated avatars that represent absent members by speech and text messages. The reactions of the avatars are determined by the previously specified ratings of the corresponding users. This simple interaction model abstracts from the complexity of the ratings on multiple dimensions (e.g., room facilities, hotel facilities, sports facilities, leisure activities) and provides awareness of other user ratings during the rating process.

All three group recommender systems have different underlying algorithms that are hard-coded. In contrast to these systems, the *GroupRecoPF* provides interchangeable merging strategies in recommendation sessions during runtime and allows their exploration as described below.

### C. Distributed Parallel Event Notification Infrastructures

The technological concepts of the *GroupRecoPF* platform are related to event notification infrastructures. The concepts benefit from existing knowledge of data handling, of distribution, and of scalability in early and established infrastructures.

Traditional event notification infrastructures (e.g., *Khronika* [17], *Elvin* [4], *NESSIE* [22], *ENI* [9]) realise the capturing, processing, storing, and delivery of data gathered from the environment. They provide interfaces for connecting clients as well as for communicating with environment components.

The *SensBution* [8] platform presents a flexible distributed approach for a sensor-based peer-to-peer architecture. Using inference rules, users query relevant information by describing the content of events they are interested in. The platform relies on a network of peers to retrieve the requested event data.

The *PPPSpace* [7] provides scalable concepts for a media space with permanent audio and video connections into distant locations as well as awareness information on events that result from various sensors. It focuses on the parallel delivery of notifications according to user preferences.

All these approaches offer mechanisms for gathering, processing, and distributing event data. However, they focus on content-filtering of event data, rather than on collaborative filtering for notifying interested clients.

## III. GROUPRECOPF CONCEPTS

The *GroupRecoPF* platform provides concepts for the flexible exploration of merging strategies, management of requests in the recommendation generation process, and programming interfaces for connecting clients.

### A. *Merging Strategy Exploration*

The *GroupRecoPF* platform relies on merging strategies for the generation of group recommendations.

It generates either group recommendations in the form of movies with the best group predictions for a given group with given constraints (i.e., circumstances that have influence on the relevancy of a movie, such as local movie show times, or the group members' budgets), or companion recommendations as other users for a given movie with given constraints (i.e., circumstances that have influence on the relevancy of companions, such as their presence on the friend list and as their movie taste). A group prediction is a given movie's anticipated overall rating for a given group. Group predictions originate from merging lists of user predictions. A User prediction is a given movie's anticipated rating for a given single user (where typically the given movie has not yet received a rating of the given user). In the *GroupRecoPF* platform user predictions are gathered from a single-user recommender based on collaborative filtering.

Merging strategies are key to the acceptance of recommendations in group recommender systems. They are often referred to as social value functions emphasising the fact that they need to find a delicate balance of the group members' diverse tastes and preferences, multifarious social relations among them, and expectations of the group activity to be recommended. Therefore, the research on merging strategies for group recommenders and their effects on the acceptance are highly explorative.

The merging strategy exploration concept of the *GroupRecoPF* platform consequently allows developers to easily explore merging strategies and their implications for the group process. The *GroupRecoPF* platform offers the *GroupRecoPF* Editor—a graphical editor that allows switching merging strategies and inspecting and manipulating their parameters at runtime. It also allows evaluators to prepare group recommender sessions for conducting user studies with a group recommender system based on the *GroupRecoPF* platform.

While any number of merging strategies can basically be applied, the *GroupRecoPF* platform currently provides three merging strategies. The two most wide-spread and well-documented merging strategies for movie recommendations: the *Weighted Maximum Average* [1, 13], and the *Weighted Maximum Minimum* [20], and one merging strategy for companion recommendations: the *Maximum Maximum*.

Table 1 presents an overview of these strategies and introduces a formal notation for describing their algorithms. Here we use the following notation:

- *M* represents the set of movies that are available to the group, with $m_j$ as a specific movie therein.
- *U* denotes the set of users in a group, with $u_i$ as an individual group member.
- The generated user prediction for a given movie and a given user is denoted as the function $p(u_i, m_j)$.
- The group weight factor $w_{ui}$ for a user $u_i$ describes the user's influence in the group recommendation generation.
- *F* denotes the set of users on the friend list of a given user, with $f_i$ as a specific friend.

| **Merging Strategies** | | |
|---|---|---|
| **Name** | **Formalisation** | **Description** |
| *Weighted Maximum Average* | $\underset{m_j \in M}{\arg\max} \left\{ \sum_{u_i \in U} \omega_{u_i} \cdot p(u_i, m_j) \right\}$ *with* $\omega_{u_i} = \frac{w_{u_i}}{\sum_{w_{u_k} \in W} w_{u_k}}$ | Generates group predictions by means of a weighted average of all user predictions. The movie with the maximal group prediction becomes the group recommendation. Has the best overall satisfaction for the group, but risk of leaving individual members behind. |
| *Weighted Maximum Minimum* | $\underset{m_j \in M}{\arg\max} \left\{ \min_{u_i \in U} \left( {w_{u_i}}^{-1} \cdot p(u_i, m_j) \right) \right\}$ | Generates group predictions by selecting the minimal user prediction of all user predictions for each movie. The movie with the maximal group prediction becomes the group recommendation. Helps individuals with minority taste, but does not optimise overall group satisfaction. |
| *Maximum Maximum* | $\underset{f_i \in F}{\arg\max} \left\{ \max_{m_j \in M} \left( p(f_i, m_j) \right) \right\}$ | Generates group predictions by selecting the maximal user prediction of all user predictions for each movie. The user with the maximal individual predictions becomes the companion recommendation. The group is suggested according to the best overall satisfaction. |

Table 1. Overview of the currently available merging strategies of the *GroupRecoPF* platform.

## B. Request Management

Efficient request management is essential for group recommender systems, since they are typically request-intense due to the fact that they are based on single-user collaborative filtering services and merging their results. The request management of the *GroupRecoPF* platform optimises the request handling between its server and its clients as well as external single-user collaborative filtering services through parallelisation and caching.

In the *GroupRecoPF* platform a central server receives numerous requests from end-users' clients who want to receive group recommendations. The server translates these requests into multiple user prediction requests that are sent to single-user collaborative filtering services. It then merges the results and sends them back to the clients.

The twofold request management allows faster computation, merging, and delivery of recommendations to clients:

- Client requests are based on session management for fast response times.
- Service requests are based on caching mechanisms for efficient data fetching.

*Client request management.* In the *GroupRecoPF* platform each recommendation process ranging from the specification of group members, movies, merging strategies and their parameters to the final delivery of the generated group recommendation is encapsulated into a unique session container. Consequently, direct and parallel access to any data of each container is possible (e.g., the group members participating in a specific recommender process, the user predictions of each individual member). The session container stores static and computed data persistently. Static data is comprised of a list of users, a list of movies, as well as the used merging strategy and its parameter configuration. Computed data is a list of movies sorted by their group predictions, out of which the group members can choose the movie to be watched. The *session history* allows access to current and past recommender sessions by end-users and by the system. End-users can browse through their personal history of sessions and reuse previously specified data. The persistence of computed data avoids performance-intense recalculations for the system. The session history is also used by evaluators, who want to evaluate the recommendation process after run-time. They can retrieve, review, and assess the recommendations in relation to the specific merging strategies and their configurations. The *session mobility* allows users to seamlessly switch between different clients with full access to session data.

*Service request management.* The *GroupRecoPF* platform optimises requests from the server to the single-user collaborative filtering service through parallelisation and caching. Parallelisation through threads allows simultaneous access to multiple external data sources at a time; the amount of threads the group recommender server used for fetching data can be configured via parameters and is between two and $M \times U$. The caching mechanism stores all data during the recommendation process about users (i.e., names, ages, profile images), relations between users (i.e., friend connections), movies (i.e., titles, first show times, plots, associated genres, production years, average community and critics ratings), user ratings (i.e., a movie's ranking as expressed opinion of a user), and user predictions (i.e., a movie's anticipated rating for a single user). Each cache has a pre-defined size of elements it can hold for later access. We distinguish two different caching types: time-relevant cache for data with an expiry time, and lifetime cache for data with permanent validity.

| **Data** | **Type** | | **Caching** |
|---|---|---|---|
| | **Time-r.** | **Lifetime** | |
| Users | | X | - |
| Friends | X | | 1d |
| Ratings | | X | - |
| Predictions | X | | 15m |
| Movies | | X | - |

Table 2. Caching parameters.

Table 2 shows an overview of data and caching times. The process of retrieving data is as follows: the server first fetches data from the caches. The caches are self-administrating: They check if the requested data is available in memory and not expired, and return the data. Otherwise, they fetch the data from the external services, store them into the cache, and return them.

## C. Programming Interfaces

The *GroupRecoPF* platform provides advanced group recommendation services, not only to its own clients, but also to third party clients via open interfaces. The interfaces support a variety of clients ranging from rich graphical desktop applications to lightweight mobile clients. While the desktop applications require convenient interfaces for accessing extensive detailed information, mobile clients require payload-optimised interfaces for accessing aggregated information. Clients that implement the end-user interface and interaction connect to the platform using either of two protocols: *XML-RPC*, or *REST/JSON*. The communication is bi-directional and allows both the access to recommender data, as well as the manipulation of recommender sessions and configurations.

The open interfaces provide full access to the group recommender and to merging strategies. They encapsulate the complexity of the recommendation generation through the sessions that contain information about: groups, their members, and available movies. Clients connected through the interfaces benefit from the previously introduced technical concepts for request management.

Third party clients can follow the whole recommendation process and have full access to all data from the *GroupRecoPF* platform. At the beginning they can initialise a group recommender process. They can retrieve information on the circumstances such as friend lists, or show times from nearby cinemas. Then, they can send data such as a list of movies, a group of users, a merging strategy and its parameters and request a group recommendation from the platform.

## IV. GROUPRECOPF IMPLEMENTATION

The software architecture of the *GroupRecoPF* platform consists of the *GroupRecoServer*, the *GroupRecoEditor*, and the *GroupRecoClients*, as well as external single-user collaborative filtering services in the *SingleUserRecoService*. Figure 1 shows the component diagram of the systems described in the subsequent sections.

Basically, the concepts for merging strategy exploration are implemented in the *GroupRecoServer* and *GroupRecoEditor* systems: the first one provides the infrastructure for switching merging strategies and configuring parameters within group recommendation sessions; the second one is responsible for allowing evaluators to configure recommendation sessions via a graphical user interface. The concepts for request management and the concepts for the programming interfaces are implemented in the *GroupRecoServer*. These concepts are used in the *GroupRecoServer's* communication with the *GroupRecoClients* and the *GroupRecoEditor*, and are provided to external services.

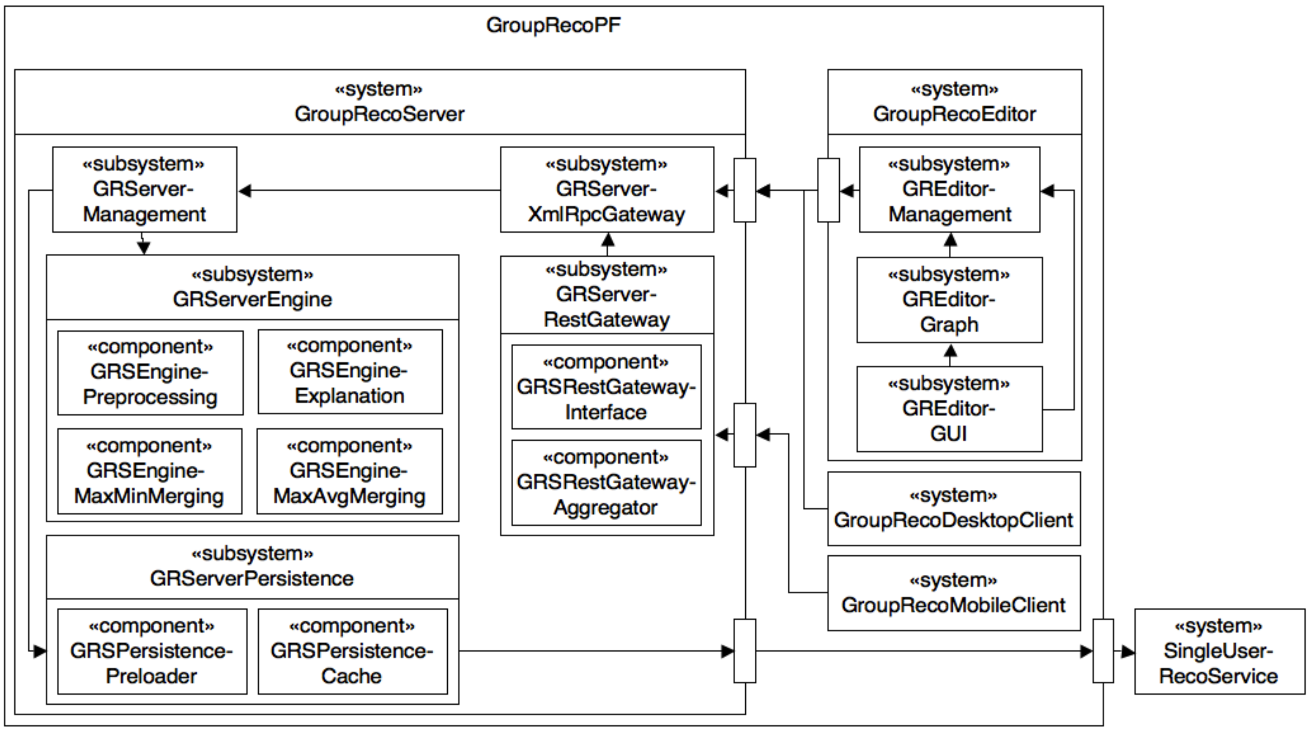
The whole *GroupRecoPF* platform including all its systems, sub-systems, and components is implemented in Java (Java 2 Platform, Standard Edition 1.6.0_20) [27] on Mac OS X 10.6.4. During the development of the platform, we repeatedly conducted white-box and black-box software tests for the implemented concepts using the JUnit 4.8.2 test suite [15].

### A. *GroupRecoServer*

The *GroupRecoServer* system provides recommendation services to all clients connected via the *GRServerXmlRpcGateway* and *GRServerRestGateway* subsystems. The *GRServerManagement* subsystem delegates requests for recommendations to the *GRServerEngine* and requests for other data to the *GRServerPersistence* subsystems. It also handles the session management with all necessary relations between movies, users, groups, cinemas, and merging strategies. The open interfaces concept is implemented in the *GRServerXmlRpcGateway* and the *GRServerRestGateway* subsystem.

The flexible merging strategies concept is implemented in the *GRServerEngine* subsystem. The implementation of the different merging strategies follows the behavioural strategy pattern [5] and allows the easy implementation and extension of respective strategies. We implemented three merging strategies, introduced above, in this subsystem. The *GRSEngineMaxMinMerging* and the *GRSEngineMaxAvgMerging* components implement two movie recommendation merging strategies; and the *GRSEngineMaxMaxMerging* component implements the companion recommendation merging strategy. They are derived from the *GroupRecoPFGREngine* abstract class and are interchangeable within the platform (cf. Figure 2).

In the *GRServerEngine* subsystem, the *GRSEnginePreprocessing* component prepares the data by fetching user predictions from the external service and


GroupRecoPF
«system» GroupRecoServer
«subsystem» GRServer-Management
«subsystem» GRServer-XmlRpcGateway
«subsystem» GRServerEngine
«component» GRSEngine-Preprocessing
«component» GRSEngine-Explanation
«component» GRSEngine-MaxMinMerging
«component» GRSEngine-MaxAvgMerging
«subsystem» GRServer-RestGateway
«component» GRSRestGateway-Interface
«component» GRSRestGateway-Aggregator
«subsystem» GRServerPersistence
«component» GRSPersistence-Preloader
«component» GRSPersistence-Cache
«system» GroupRecoEditor
«subsystem» GREditor-Management
«subsystem» GREditor-Graph
«subsystem» GREditor-GUI
«system» GroupRecoDesktopClient
«system» GroupRecoMobileClient
«system» SingleUser-RecoService
Figure 1. Component diagram of the *GroupRecoPF* platform.

aggregating them. The *GRSEngineExplanation* component logs crucial decisions of the system during the recommendation process. This information is presented to the group in the form of explanations. We have implemented an abstract explanation class (i.e., containing a rating value, a reference to a certain user or a reference towards a particular movie) and derived eight specialised explanation types from it.

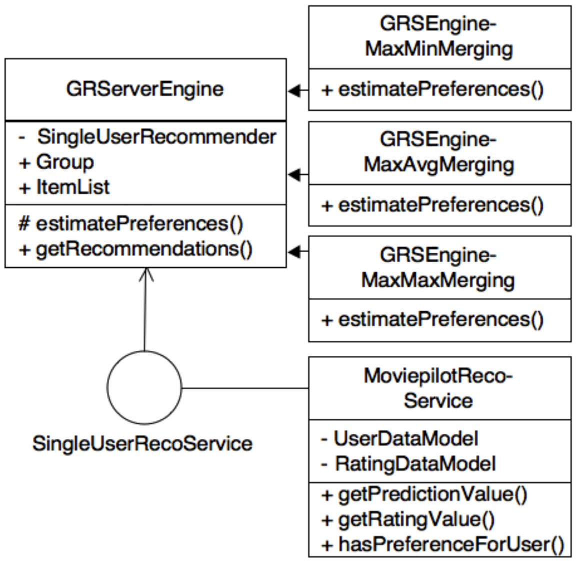


Figure 2. Class diagram of the merging strategies following the strategy pattern.

The *GRServerPersistence* subsystem implements the parallel data retrieval and caching mechanisms as introduced before. The *GRSPersistencePreloader* provides configurable thread implementations (i.e., `SwingWorker`) that allow the preloading of relevant user and group data, such as the list of friends.

The *GRSPersistenceCache* component implements the cache mechanisms, which use a generic constructor `Cache(Retriever<super K, extends V> retriever)` with a `Retriever` instance as argument. The retrievers are specialised for the corresponding data objects. They obtain the value for a given key, if no cache value exists. A `FastMap<K, V>` is used as underlying data structure in the *GRSPersistenceCache*. Synchronised access to the `put(K,V)` and `get(K)` methods guarantee consistency. For time-relevant data, such as predictions, we implemented specialised caches with a second `FastMap<K, Calendar>` data structure that is responsible for holding the last update time for an associated key. A date comparison using the `Calendar` data structure determines if either the value received from the cache is still valid, or the value should be obtained using a `Retriever`. All caches are configurable in size and expiry time via the platform's properties.

## B. *GroupRecoEditor*

The *GroupRecoEditor* provides a graphical front-end to the *GroupRecoPF* platform, and is implemented as a specialised *GroupRecoDesktopClient* that connects to the group recommender system using the *GRServerXmlRpcGateway* subsystem. As shown in Figure 1, the *GroupRecoEditor* system is composed of three subsystems: *GREditorManagement*, *GREditorGUI*, and the *GREditorGraph*.

The *GREditorManagement* subsystem consists of components for the management and delegation of the core subsystems and of components of the *GroupRecoEditor* system. It is responsible for the creation and management of graph nodes in the *GREditorGraph* subsystem and relies on data from the *GroupRecoServer*. For the frequent access to movie poster images and user profile images, the *GREditorManagement* subsystem provides the `PosterImageBuffer` and `UserImageBuffer` components, which retrieve and cache image data. Adding new merging strategies to the *GroupRecoEditor* requires little effort for changes in the *GREditorManagement* subsystem.

The *GREditorGUI* subsystem manages all graphical user interface elements and is responsible for the user interaction with the system. It represents the application's main window, and contains and manages all other user interface elements. Users can instantiate components and request details about users, movies, and merging strategies. Generated recommendations from the *GroupRecoServer* are displayed, together with the corresponding explanations. The *GREditorGUI* subsystem provides a set of different implementations of `JGlassPane` objects for advanced visual effects and user interaction techniques. It also makes use of the *Quaqua* [23] and the *Mac Widgets for Java* [21] frameworks.

The *GREditorGraph* subsystem provides a graph model that encapsulates information about active nodes, edges, and metadata and is based on the `JGraph` [14] component. It also realises the edges according to advanced spline-based routing.

## C. *GroupRecoClients*

The *GroupRecoClients* realise group interaction with specialised graphical user interfaces, according to different application scenarios. In the current implementation the *GroupRecoPF* platform provides two types of clients: *GroupRecoMobileClient*, and *GroupRecoDesktopClient*.

The *GroupRecoMobileClient* is a mobile client for the Apple iPhone and supports group agents in finding a movie for their groups. It connects to the *GroupRecoServer* via the *GRServerRestGateway*, implemented in `REST/JSON` [2, 3]. The *GRSRestGatewayAggregator* prepares the data in the JavaScript Object Notation (`JSON`) format, the *GRSRestGatewayInterfaces* provide API calls for clients to access and manipulate data, such as configuring group recommendation sessions.

*GroupRecoDesktopClients* such as the *GroupRecoEditor* provide rich graphical applications for desktop computers. They typically connect to the *GroupRecoServer* via the *GRServerXmlRpcGateway*, implemented in XML Remote Procedure Call (XML-RPC) [26].

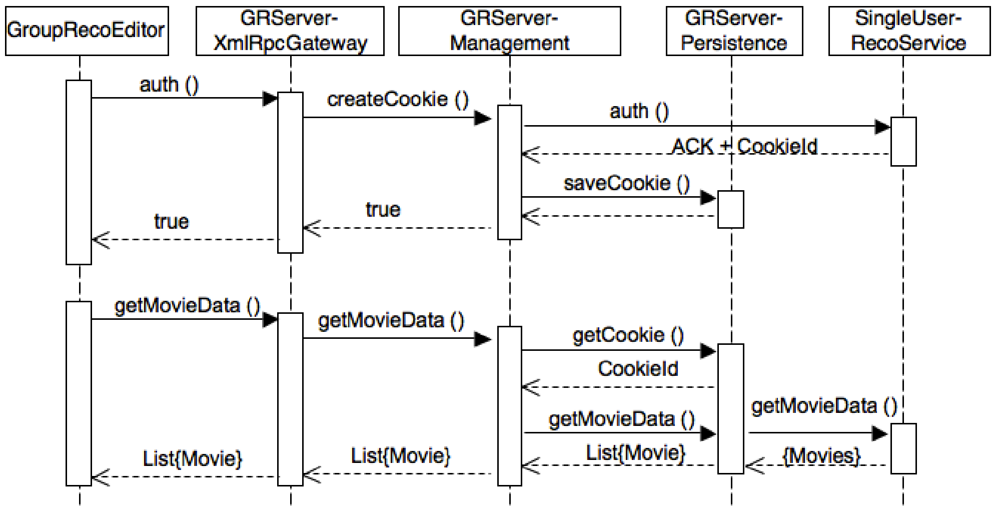


Figure 3. Sequence diagram of a typical *GroupRecoEditor* request towards the *SingleUserRecoService* for retrieving movie data.

## *D. SingleUserRecoService*

The *SingleUserRecoService* provides interfaces and implementations for retrieving data about movies, users, ratings, and predictions. In the current implementation, it is provided through an external single-user collaborative filtering service.

For guaranteeing a high level of security, all requests for data are encapsulated in user-authenticated sessions, which are handled in the various subsystems and passed by all subsystems beginning with the clients. The communication between the *GroupRecoPF* and the *SingleUserRecoService* is based on an `REST/JSON` API.

Figure 3 shows a typical client request (e.g., initiated in the *GroupRecoEditor* system) for retrieving movie data. First, the users authenticate in the editor. Then, the *GRServerXmlRpcGateway* subsystem handles this request and distributes it within the *GRServerManagement* subsystem. The *GRServerManagement* is responsible for the session handling. After successfully authenticating against the *SingleUserRecoService*, it creates a local cookie. This cookie is stored in the *GRServerPersistence* subsystem for later use.

The second request in Figure 3 retrieves movie data and relies on the previously created cookie for authentication. The *GroupRecoEditor* initiates the request, and the gateway subsystem forwards the request to the *GroupRecoManagement* subsystem. The *GroupRecoPersistence* subsystem delivers the requested cookie, which is required to retrieve data from the *SingleUserRecoService*. As every cookie has an expiry time, the validity of the cookie is checked. The *SingleUserRecoService* returns the movie data in the `JSON` format. This data is finally converted for the XML-RPC protocol and transferred to the requesting *GroupRecoEditor*. In case the authentication fails, the system notifies the requesting client.

## V. CONCLUSIONS

In this paper we introduced the *GroupRecoPF*. It provides flexible merging strategies and facilitates their exploration. It offers parallel processing of requests for non-blocking access to the system's components, and for multi-threaded data fetching from external data sources, as well as persistent storing of recommendation sessions for later reuse, and programming interfaces providing gateways to clients. So far, the *GroupRecoPF* platform is completely implemented and deployed and has been used for eight weeks.

For evaluating the accuracy and the acceptance of group recommendations generated by new merging strategies the *GroupRecoPF* platform can easily be extended. The *GroupRecoPF* platform provides the *GroupRecoEditor* for developers and researchers. The editor's graphical user interface dynamically adapts to new merging strategies and shows their parameters. It also allows evaluators to compare data of various recommendation sessions, as well as manipulate sessions during group studies for evaluating the social implication of parameters. We tested the *GroupRecoPF* platform with an attached *GroupRecoMobileClient* with fifteen users, organised in teams of three users each.

Future work remains in measuring the performance of the platform for distribution and scalability, and evaluating the acceptance of the given recommendations in user studies. It would be interesting to test the performance of the platform with varying numbers of teams (e.g., ten, twenty, thirty teams), sizes (e.g., three, seven, ten members), hardware (e.g., desktop PCs, smartphones, multi-touch devices) and network connectivity (e.g., 2G, 3G, WiFi), and circumstances (e.g., low, medium, high availability of movies and cinemas in neighbourhood). While the *acceptance* of the two merging strategies for group recommendations of movies has been well-documented, the experience with the companion recommendation merging strategy is limited. Acceptance evaluations should, therefore, address companion recommendations.

Also, extending the programming interfaces for multifarious communication among the platform's systems would be interesting (e.g., facilitating communication among the clients in situations were wide area networks are not available using Bluetooth).

## ACKNOWLEDGEMENTS

We thank all members of the Cooperative Media Lab, and the moviepilot GmbH. Part of the work has been funded by the German Research Foundation (DFG GR 2055/2-1). Thanks to the anonymous reviewers.

## REFERENCES

[1] Ardissono, L., Goy, A., Petrone, G., Segnan, M. and Torasso, P. Intrigue: Personalised Recommendation of Tourist Attractions for Desktops and Handset Devices. Applied Artificial Intelligence - An International Journal 17, 8-9 (2003). pp. 687-714.

[2] Crockford, D. JSON: JavaScript Object Notation. http://www.json.org/, 2010. (Accessed 2/8/2010).

[3] Fielding, T. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. thesis, Don Bren School of Information and Computer Science, University of California, Irvine, Irvine, CA, 2000.

[4] Fitzpatrick, G., Mansfield, T., Kaplan, S., Arnold, D., Phelps, T. and Segall, B. Augmenting the Workaday World with Elvin. In Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work - ECSCW'99 (Sept. 12-16, Copenhagen, Denmark). Kluwer Academic Publishers, Dortrecht, NL, 1999. pp. 431-450.

[5] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 1994.

[6] Goldberg, D., Oki, B., Nichols, D. and Terry, D.B. Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM 35, 12 (Dec. 1992). pp. 61-70.

[7] Gross, T., Beckmann, C. and Schirmer, M. The PPPSpace: Innovative Concepts for Permanent Capturing, Persistent Storing, and Parallel Processing and Distributing Events. In Proceedings of the Eighteenth Euromicro Conference on Parallel, Distributed, and Network-Based Processing - PDP 2010 (Feb. 17-19, Pisa, Italy). IEEE Computer Society Press, Los Alamitos, 2010. pp. 359-366.

[8] Gross, T., Paul-Stueve, T. and Palakarska, T. SensBution: A Rule-Based Peer-to-Peer Approach for Sensor-Based Infrastructures. In Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications - SEAA 2007 (Aug. 27-31, Luebeck, Germany). IEEE Computer Society Press, Los Alamitos, 2007. pp. 333-340.

[9] Gross, T. and Prinz, W. Modelling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. Computer Supported Cooperative Work: The Journal of Collaborative Computing 13, 3-4 (Aug. 2004). pp. 283-303.

[10] Han, P., Xie, B., Yang, F. and Shen, R. A Scalable P2P Recommender System Based on Distributed Collaborative Filtering. Journal of Expert Systems with Applications 27, 2 (Aug. 2004). pp. 203-210.

[11] Herlocker, J.L., Konstan, J.A., Terveen, L. and Riedl, J. Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems 22, 1 (Jan. 2004). pp. 5-53.

[12] Jameson, A. More Than the Sum of Its Members: Challenges for Group Recommender Systems. In Proceedings of the Working Conference on Advanced Visual Interfaces - AVI 2004 (May 25-28, Gallipoli, Italy). ACM, N.Y., 2004. pp. 48-54.

[13] Jameson, A. and Smyth, B. Recommendation To Groups. In Brusilovsky, P., Kobsa, A. and Nejdl, W., eds. The Adaptive Web. Springer-Verlag, Heidelberg, 2007. pp. 596-627.

[14] JGraph Ltd. Java, AJAX, and Flash Graph Visualisation and Layout. http://www.jgraph.com/, 2010. (Accessed 2/8/2010).

[15] JUnit.org. Welcome to JUnit.org! | Junit.org. http://www.junit.org/, 2010. (Accessed 2/8/2010).

[16] Lieberman, H., Van Dyke, N.W. and Vivacqua, A.S. Let's Browse: A Collaborative Web Browsing Agent. In Proceedings of the 4th International Conference on Intelligent User Interfaces - IUI'99 (Jan. 5-8, Los Angeles, CA). ACM, N.Y., 1999. pp. 65-68.

[17] Loevstrand, L. Being Selectively Aware with the Khronika System. In Proceedings of the Second European Conference on Computer-Supported Cooperative Work - ECSCW'91 (Sept. 24-27, Amsterdam, NL). Kluwer Academic Publishers, Dortrecht, NL, 1991. pp. 265-278.

[18] McCarthy, J.F. and Anagnost, T.D. MUSICFX: An Arbiter of Group Preferences for Computer-Supported Collaborative Workouts. In Proceedings of the ACM 1998 Conference on Computer-Supported Cooperative Work - CSCW'98 (Nov. 14-18, Seattle, WA). ACM, N.Y., 1998. pp. 363-372.

[19] Monroe, D. Just For You - Recommender Systems That Provide Consumers with Customised Options have Redefined e-Commerce, and are Spreading to other Fields. Communications of the ACM 52, 8 (Aug. 2009). pp. 15-18.

[20] O'Connor, M., Cosley, D., Konstan, J.A. and Riedl, J. PolyLens: A Recommender System for Groups of Users. In Proceedings of the Seventh European Conference on Computer-Supported Cooperative Work - ECSCW 2001 (Sept. 16-20, Bonn, Germany). Kluwer Academic Publishers, Dortrecht, NL, 2001. pp. 199-218.

[21] Orr, K. macwidgets - Project Hosting on Google Code. http://code.google.com/p/macwidgets/, 2010. (Accessed 2/8/2010).

[22] Prinz, W. NESSIE: An Awareness Environment for Cooperative Settings. In Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work - ECSCW'99 (Sept. 12-16, Copenhagen, Denmark). Kluwer Academic Publishers, Dortrecht, NL, 1999. pp. 391-410.

[23] Randelshofer, W. Quaqua Look and Feel. http://www.randelshofer.ch/quaqua/, 2010. (Accessed 2/8/2010).

[24] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of the Conference on Computer-Supported Cooperative Work - CSCW'94 (Oct. 22-26, Chapel Hill, NC). ACM, N.Y., 1994. pp. 175-186.

[25] Resnick, P. and Varian, H.R. Introduction to special issue on Recommender Systems. Communications of the ACM 40, 3 (Mar. 1997). pp. 56-58.

[26] Scripting News Inc. XML-RPC Home Page. http://www.xmlrpc.com/, 2010. (Accessed 2/8/2010).

[27] Sun Microsystems Inc. J2SE 6.0. http://java.sun.com/javase/6, 2010. (Accessed 2/8/2010).

[28] Terveen, L., Hill, W., Amento, B., McDonald, D. and Creter, J. PHOAKS: A System for Sharing Recommendations. Communications of the ACM 40, 3 (Mar. 1997). pp. 59-62.

[29] Vozalis, E. and Margaritis, K.G. Analysis of Recommender Systems' Algorithms. In Proceedings of the 6th Hellenic European Conference on Computer Mathematics and its Applications - HERCMA 2003 (Sept. 25-27, Athens, Greece). Athens University, Athen, Greece, 2003. pp. 732-745.