MASTERY GRIDS
MANUAL

DEVELOPED BY

PAWS LAB

UNIVERSITY OF PITTSBURGH

08/13 ~ 03/14

OUTLINE

Introduction

Mastery Grids is a visual-rich, interactive, adaptive E-learning platform with integrated functionalities enabling multi-facet social comparison, open user modeling, and multi-type learning materials support. It presents and compares user learning progress and knowledge level (mastery) by colored grids, tracks user activities and feedbacks dynamically and provides flexible user-centered navigation across different content levels (e.g. topic, question) and different content types (e.g. question, example) of learning materials.

The architecture supporting **Mastery Grids** fulfills a major objective, portability, which is the ability to be integrated to other systems with little set up and modification. The architecture is modular and includes different software components:

- frontend user-system interaction interface, the **Mastery Grids interafce**,
- backend **Aggregate services** communicating between the main interface and user modeling services,
- backend **user modeling services**, and
- backend **content providing applications**.

Aggregate services deliver user mastery information to the frontend interface after aggregating information from **user modeling services**. The aggregation is performed on both individual level and group level (course class), lower content level (e.g. question) and higher content level (e.g. topic). The services also track user actions(interactions) on Mastery Grids interfaces into aggregate database.

The **content providing applications** are loaded from the frontend interface by simple urls, and are mean to report within application up-to-date user activities to user modeling services.

Overall view of the architecture
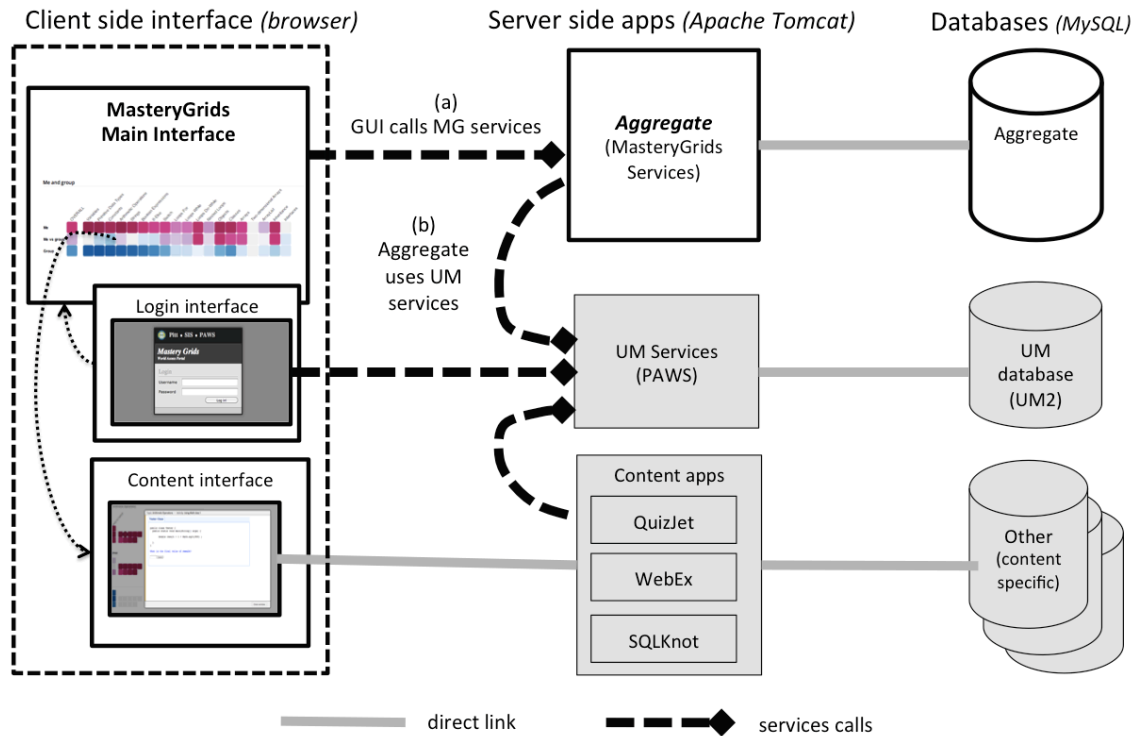
The above diagram gives a graphical explanation of the overall architecture. Diagram disposition is in columns.

- First column represents the client side (browser) displaying the *Mastery Grids* Interface(s). A login screen has been implemented (cbum login), and it uses services to authenticate users in the PAWS User Model. Contents (questions/examples) are displayed in an overlay window with a popout *iframe* that loads other applications (like Quizjet for java questions, or Webex for examples).
- Second column represents server side applications (currently all of them are in adapt2 server running apache Tomcat). Aggregate is web application built on Java containing services to feed information to Mastery Grids Interface based on the Mastery Grids Protocol (JSON format). PAWS UM services is a set of web application providing access to PAWS User Model functionalities (e.g compute levels of knowledge per concept, authenticate, provide activity records of the user, etc.). Content apps represent different web applications providing content (like examples and java questions).
- Third column represents the databases. All of them are MySQL.

The main interactions of Mastery Grids are listed in points (a) and (b):
  a) MG GUI calls Aggregate services for i) getting content definitions, course structure (questions, examples, topics) and progress/knowledge levels (user model), as defined in the MG Protocol; and ii) tracking student interaction (logs).
  b) MG service calls UM services to get activity records and concept levels of knowledge of the student and compute aggregated knowledge/progress per content and per topic.

Detailed View of Mastery Grids Components
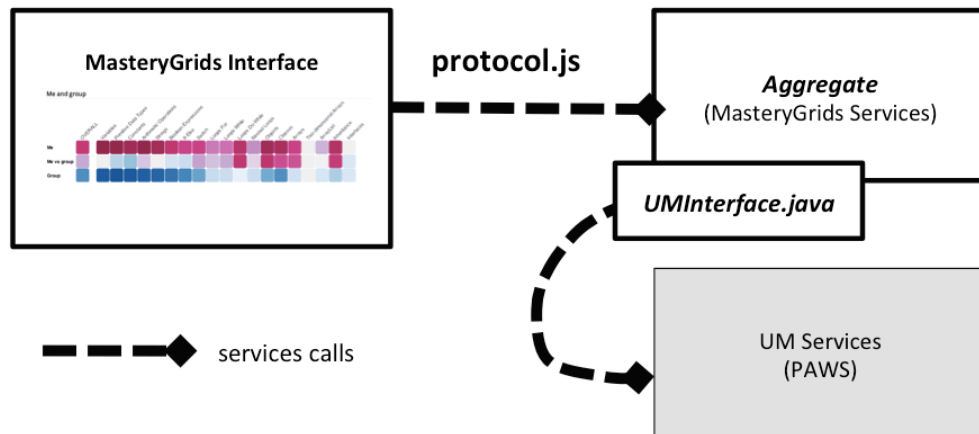
The above diagram shows the detailed view of *Mastery Grids* Components. The interaction between Aggregate and PAWS UM Services is defined in the java interface *UMInterface*. This interface is implemented in the class *PAWSUMInterface.java*, where the services for getting information from the PAWS User Model are defined as a working example. To connect to another user model, *UMInterface.java* has to be implemented with a proper class. More details will be provided in chapter **The Services.**

Installation (for developers)

This chapter provides guidance to install Mastery Grids interface and Aggregate services and databases onto your designated server. By default, all the code uses PAWS Lab server (http://adapt2.sis.pitt.edu) and services (applications) to provide Aggregate services, user modeling services and content providing applications. If you plan to use these default settings, feel free to skip this chapter. Depending on your needs, you can choose to skip some steps:

- For using default Aggregate services and databases, skip step 2;
- For using default user modeling services and databases, skip step 3;
- For using default content providing applications, skip step 4.

Here are the complete steps for installation.

1. Download source codes and database from (https://github.com/PAWSLabUniversityOfPittsburgh/MasteryGrids). Place **um-vis-adl** (Mastery Grids interface) code and **aggregate** services code on the same designated server. Mastery Grids interface is a javascript application that communicates to agreagte services, and for security javascript restrictions, it can not make cross-domain calls. There are three parts in GitHub:
   - **um-vis-adl**: This contains interface source codes including protocol.
   - **aggregate**: This contains *Aggregate services* delivering user mastery information to interfaces after communication with user modeling services, and tracking user actions (etc.) into *Aggregate database*.
   - **aggregate_db:** This is *Aggregate databases*.

2. Create databases (using *aggregate_db* or your own databases) and configure database connection.
   - Set config.xml (under *aggregate/WebContent/WEB-INF*) correspondingly for database connection.
   - Increase the maximum length of the MySQL system variable group_concat_max_len. (e.g. group_concat_max_len variable = 8192). This is necessary for some of the queries in Aggregate using long texts.

3. Modify *Aggregate services* code if you use your own user modeling server. For this, implement the java interface *UMInterface.java* and change the corresponding class name in *Aggregate.java; PAWSUMInterface.java* shows an example implementation to access PAWS lab user modeling services (*aggregateUMServices*).

Aggregate.java

```
public Aggregate(String usr, String grp, String cid, String sid, HttpServlet servlet) {
  …
  public UMInterface um_interface;
  …
  um_interface = new PAWSUMInterface(); // change to your implementation
  …
}
```

PAWSUMInterface.java

```
public class PAWSUMInterface implements UMInterface{
  private String server = "http://adapt2.sis.pitt.edu";
  // not critical, only will prevent to get personal information of users
  private String secureKey = "put_here_the_key";

  private String userInfoServiceURL = server+"/aggregateUMServices/GetUserInfo";
  private String classListServiceURL = server+"/aggregateUMServices/GetClassList";
  private String questionsActivityServiceURL =
                          server+"/aggregateUMServices/GetQuestionsActivity";
  private String examplesActivityServiceURL =
                          server+"/aggregateUMServices/GetExamplesActivity";
  private String conceptLevelsServiceURL = server+"/cbum/ReportManager";
  …
}
```

4. To use your own content providing application, the URLs referring to the content items have to be properly defined in the aggregate database, where the content and topics of the courses are defined. Also, the content providing application should log information to the user model. The connection between the content providing application and the User Model is not described here, and will depend on the content providing application implementation and the user model services used.

The System (for client-side users)

In this chapter we provide guidance for using Mastery Grids system as a client-side user (e.g. student).

Entering Mastery Grids System

We provide a default log in page by http://adapt2.sis.pitt.edu/cbum/mg-adl/. You can log in with user adl01 (same password).

Main Elements in Mastery Grids Interface

The next figure dissects the Mastery Grids Visualization interface.

Configuration Bar allows to change different visualization aspect.

The topic list

Me grid shows your mastery for each topic by 3-row grids. Row 1 presents the average over different contents. Row 2 presents that in questions. Row 3 shows that in examples. Deeper red indicates higher mastery.

First column shows the average over topics.

Comparison grid compares your mastery with that of the group (see grid below). Deeper blue indicates higher mastery of group than yours, and deeper red means the opposite.

Group grid shows average of the entire class or top students. This option can be set in the *configuration bar* above.

Learners in group grid shows other students in similar way as *Me grid*.

Configuration Bar

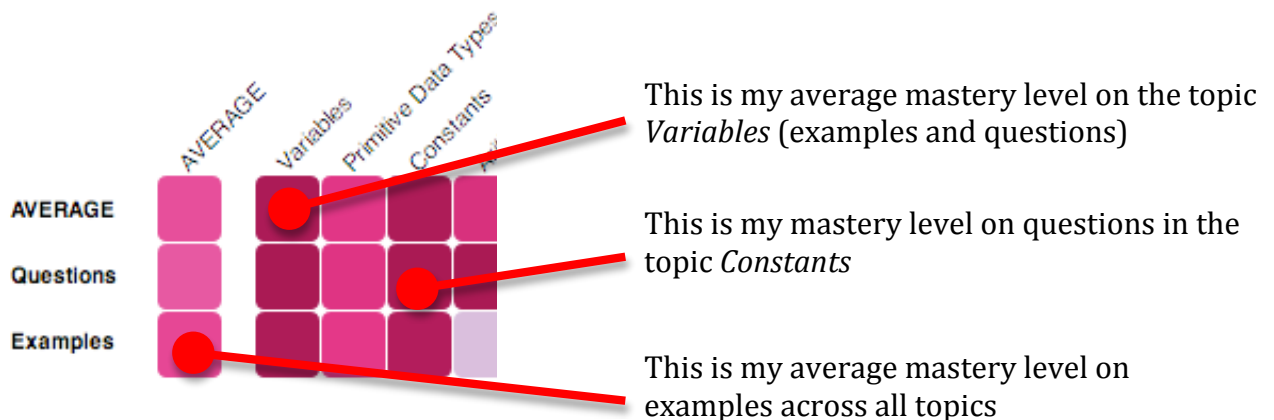The configuration bar allows you to change some aspects of the interface.

- **Comparison** (Group/individual): By default *Group* is selected. *Group* comparison enables the comparison with the average or top students group. If *individual* option is chosen, current student's grid is followed by the grids of the individual class students and the color is discarded (showing grey scale). *Individual* option is useful if you want to compare yourself with individual students. However, student names or nick names are not shown.

- ▪ **Report level** (Progress/Knowledge)**:** You can choose either showing progress or knowledge level. This aspect is optional to show on the bar.
- ▪ **Topic Size** (Equal/Difficulty/Importance): You can choose different sizes for topics according to topic difficulty or importance (need to define in the backend services). This aspect is optional to show on the bar.
- ▪ **Group** (Class Average/Top 10): By default *Class average* is set. *Class average* shows the average of all students in the class in the *Group* grid. *Top 10* option shows the average of the top 10 students in the class in the *Group grid*.
- ▪ **px:** This is the option for changing the pixel size of the cells in *Learners in group grid* (when Resources is not ALL).
- ▪ **Resources** (All/OVERALL/Questions/Examples): By default *All* is selected. This drop down list allows you to select which types of resources you want to display. For example, if you want to focus in *Questions* and want to compare only your levels of mastery on questions with the group, choose *Questions* here. The interface will reduce the grids only to show the question's rows.
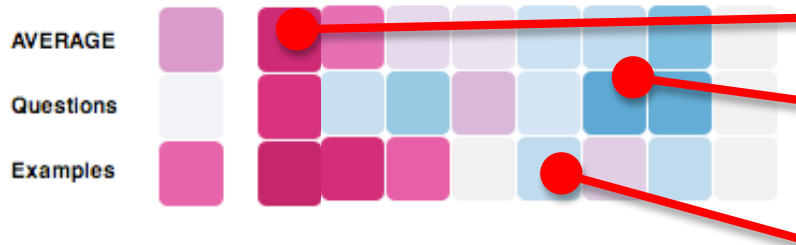
Me Grid
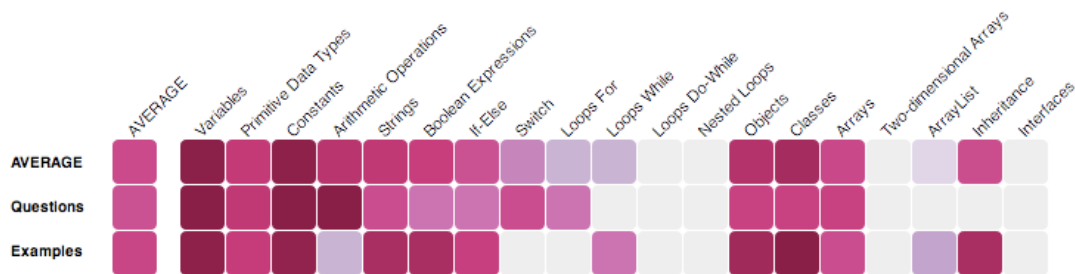
The grid shows progress in each topic, for each type of content (questions, examples) and averages across content types and across topics. The rows are: *AVERAGE*, *Questions*, and *Examples*. In the columns you can see each topic. The first column represents the average across topics. Mastery in a topic is computed by averaging the mastery on individual content (questions and/or examples) in the topic. A dark violet color means high mastery. Light grey cells indicate no progress.

This is my average mastery level on the topic *Variables* (examples and questions)

This is my mastery level on questions in the topic *Constants*

This is my average mastery level on examples across all topics

Comparison Grid

A violet colored cell indicates that you have more progress than the

A blue cell means that the group has more progress than you.

A slightly blue or violet cell indicates that the difference is little.

Group Grid

Group grid shows group average progress in the same way than *Me Grid*. To make the difference it uses color blue instead of violet.
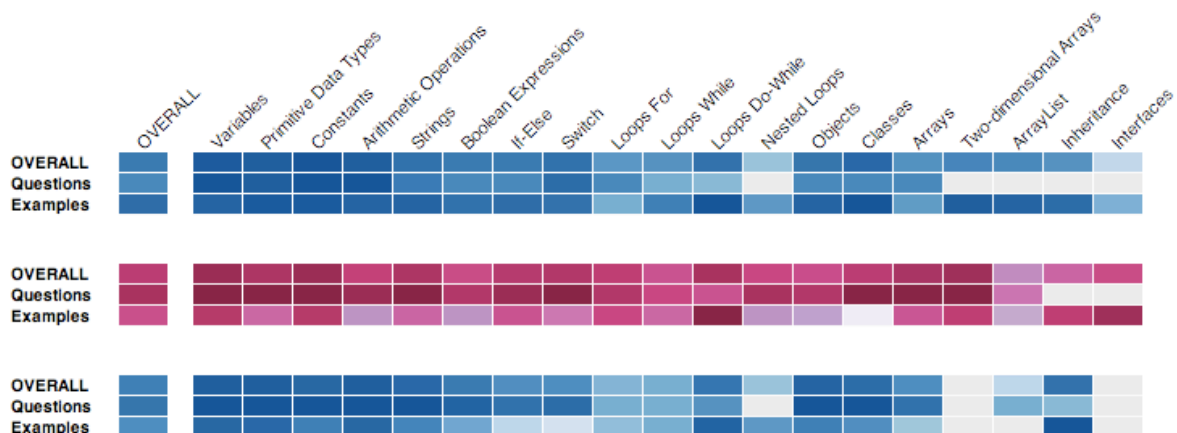

Learners in Group Grid

Learners in group grid shows one grid per student in the group. By choosing *Class average* in the *configuration bar*, all students in the class are included. By choosing *Top 10*, only top 10 students are listed (The grid only shows all the peers including current student that have logged into the system at least once).

Comparing Mastery Among Cells

Sometimes color changes are subtle and comparing levels among cells can be hard. When a cell is selected (shape rotates) a bar chart is displayed at the right side showing an alternative comparison view among the cells on the row and highlighting the selected cell in dark grey.


The progress of the selected cell is shown as a dark grey bar in the chart at the right side
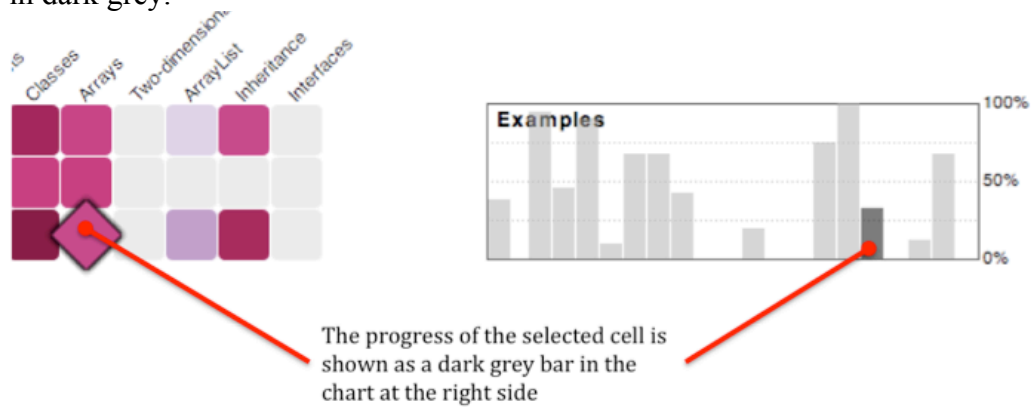
Accessing Content

The content (questions, examples) inside a topic can be accessed by clicking a cell.

Entering the topic _Loop For_
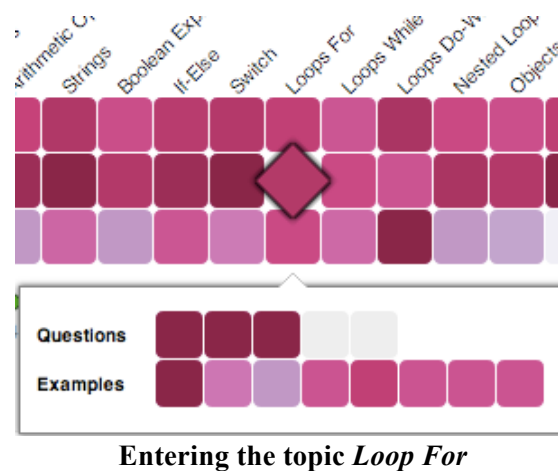
To display a content item, click on the cell. In the next figure, the first question has been clicked and it is marked as a circular cell.

As following figure shows, the **question** is loaded in an overlay window. The name of the question can be read in the top part of this window: _Loop For 1_.
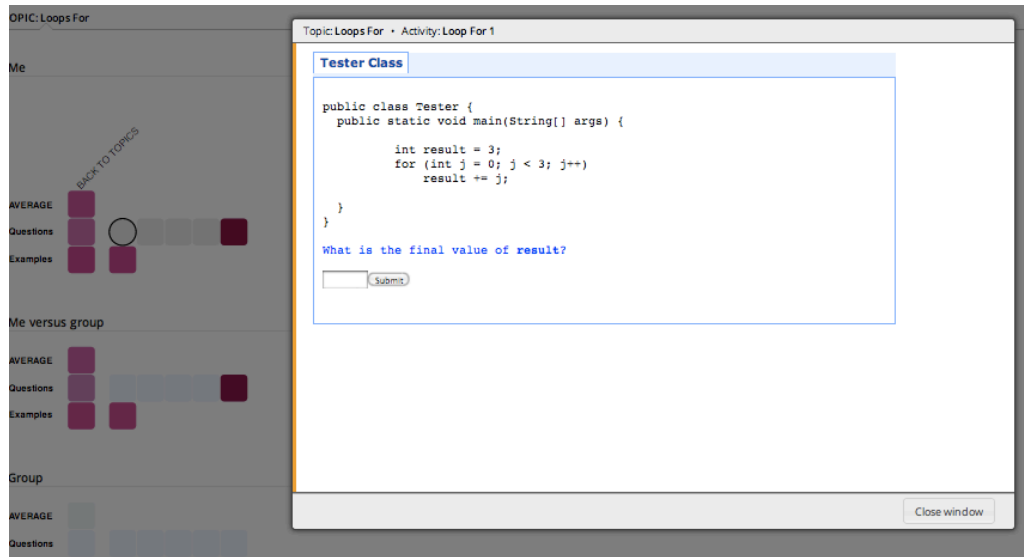
Questions

Content items of type **questions** are **exercises** in which you must give an **answer**. If your answer is correct, the mastery level in the question will change. If incorrect, you can repeat the question. Note that current system by default provides Quizjet questions, so when repeating a question some of the parameters inside the question would change, thus the correct answer would be different.

Examples

An **example** shows a piece of code and allows you to display comments explaining the important lines of code on the example. Current system by default provides WebEx examples.
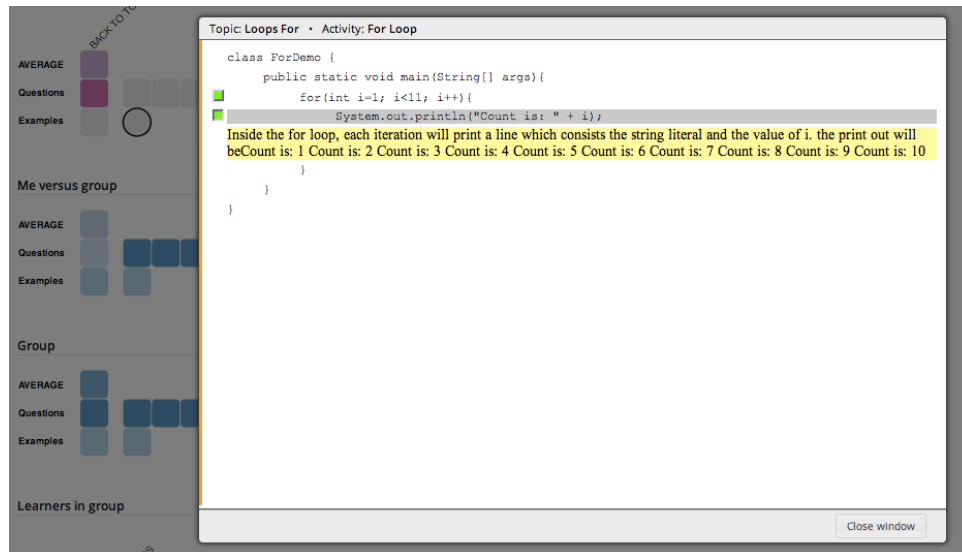
In this case, the user clicked in the example named is *For Loop* and displayed the second commented line in it.

The Protocol

The protocol is the communication bridge between *Mastery Grids* interface (MG GUI) and *Aggregate Services*. It enables i) getting content definitions, course structure (questions, examples, topics) and progress/knowledge levels (user model) from JSON returned from *Aggregate Services*, and ii) tracking student interaction (logs) (and feedbacks) by passing information in urls calling *Aggregate Services*. The protocol is desribed in details within the JSON format file ***protocol.js***. Here, a brief view of the objects defined by the protocol is provided.

- MAIN object (i.e., the response of the server)
- REPORT_LEVEL object
- FEEDBACK_FORM_ITEM object
- TOPIC object
- ACTIVITY object
- CONCEPT object
- LEARNER object
- GROUP object
- STATE object

The Services

Here we introduce more details the java codes of *Aggregate Services* which
- delivers information to MG Interface
- connects to UM services.

Following graph shows the functionalities(roles) of the main java classes.

Aggregate Services Details
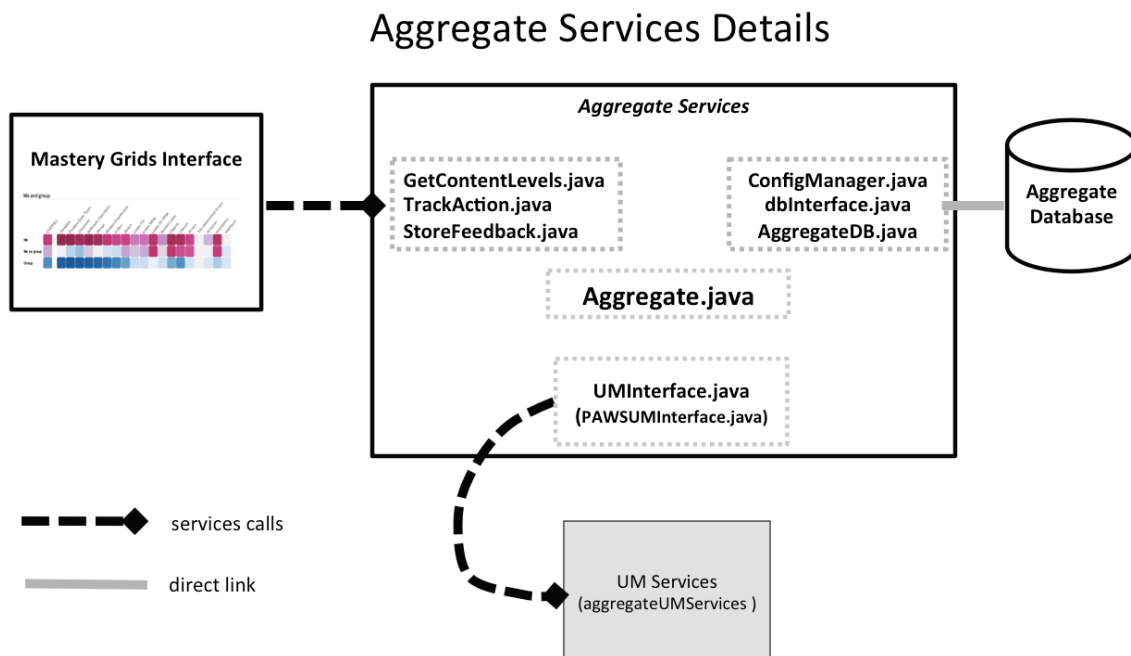
The Databases

Here we mainly introduce Aggregate Database that supports *Aggregate Services*. Following graph shows the main tables in the database. Details about the fields can be checked after downloading GitHub database files, and the connection between this database with *Aggregate Services* can be checked in *Aggregate Services* java codes.

```
▼ 🗄 aggregate
  ▼ 🗂 Tables
    ▶ 🃏 ent_content
    ▶ 🃏 ent_content_tree
    ▶ 🃏 ent_content_type
    ▶ 🃏 ent_course
    ▶ 🃏 ent_domain
    ▶ 🃏 ent_group
    ▶ 🃏 ent_hidden_topics
    ▶ 🃏 ent_jexample_concept
    ▶ 🃏 ent_jquiz_concept
    ▶ 🃏 ent_non_student
    ▶ 🃏 ent_precomputed_models
    ▶ 🃏 ent_recommendation
    ▶ 🃏 ent_recommendation_approach
    ▶ 🃏 ent_topic
    ▶ 🃏 ent_tracking
    ▶ 🃏 ent_user_feedback
    ▶ 🃏 rel_con_con_sim
    ▶ 🃏 rel_content_concept
    ▶ 🃏 rel_topic_content
    ▶ 🃏 rel_topic_course
    ▶ 🃏 sql_question_names
    ▶ 🃏 study_show_rec
```