Scripted Collaboration in MOOCS

Motivation:

In face-to-face and Computer Supported Collaborative Learning (CSCL) contexts, collaboration in small groups has been shown to provide benefits to learners [1], especially in critical thinking and reasoning oriented tasks [2]. However, it has been observed that mere collaboration itself is not sufficient to positively impact learning [3]. In fact, it can have a negative impact unless the collaboration is structured, usually by means of a script.

Despite studies [4,5] indicating the advantages of structured collaboration towards both learning outcomes and student satisfaction among older student populations as found in MOOCs, social isolation has been the norm in most MOOC platforms. One notable exception is NovoEd which makes its courses inherently collaborative, requiring students to form groups early in the course. Many courses such as the free 'Technology Entrepreneurship' course [6] on NovoEd boast completion rates as high as 65% [7] as compared to the anemic 5-10% typical of other platforms.

Encouraged by this, the DANCE group at CMU is making progress towards supporting scripted collaboration in MOOCs using two major approaches:

1. Learning Tools Interoperability (LTI):
   LTI is a standard that defines how 3rd party tools (hosted on external servers) can be used in LTI compliant MOOC platforms (called tool consumers) such as EdX, Coursera etc. In our group's work, a conversational chat tool called Bazaar [8] has been successfully embedded in EdX courses using this method.

2. Open EdX XBlock framework:
   XBlock is the modular and hierarchical component architecture used by EdX to construct courses. Each XBlock can be described as a web application in a div. In this document, we propose a preliminary design for an XBlock that can support both asynchronous and synchronous scripted collaboration.

The LTI approach is appealing because it potentially enables the use of a single tool across multiple MOOC platforms. However, this approach is reliant on each platform (tool consumer) for exposing data and services in an LTI compliant fashion. This might include information about user profiles, course progress information etc. Indeed, the standard itself may have to change to accommodate future needs.

Contrastively, the XBlock framework is tightly integrated with the EdX platform. EdX's own use of the XBlock framework guarantees availability of all features, including those released in the future, such as the cohort and team features scheduled for release later this year. While the

XBlock framework is primarily being used by EdX, it has been designed so that other platforms can make use of it as well. For example Google Course Builder uses it and will eventually use the OpenEdX stack to form www.mooc.org.

Design Considerations:

Before detailing the actual design of the XBlock, here are some aspects that have been taken into consideration while designing it:

1. Pre-collaboration activities
2. Group Formation and Role Allocation
3. Customizable User Interfaces (UIs)
4. Supporting Multiple UI types and convertibility between them
5. Targeting of UIs based on Role/Group
6. Synchronous vs. Asynchronous use cases
7. Integrating a Tutoring System
8. Integrating a static script

Design:

An XBlock can be described as a web application within an HTML div. This enables individual XBlocks to be self-contained, modular and nested in a hierarchical structure. Since XBlocks are effectively web applications in their own right, we can have a Model-View-Controller (MVC) architecture for each XBlock.

View defines the user interface that is exposed to students by the XBlock. Certain aspects of a script can be built into the view, such as scaffolding of student moves, restriction of move types etc. The View is usually specified using HTML and CSS.

Model is the internal representation of the collaborative activity in a database. Ideally the model should be invariant with respect to the specific User Interface and collaborative activity design and should expose a clearly defined API through which it can be used. Our group has already made efforts towards this in the form of DiscourseDB.

Controller is a component that has knowledge of the UI and the model (API) used. It effectively translates student actions via the UI to API calls to the model. Controller is usually implemented using JavaScript.

There are two aspects to the UI. The first is how the collaborative activity is depicted. Based on our survey of various CSCL tools, the view and controller components must be customizable to accommodate user interfaces that can be:

1. Linear:
   Synchronous chat applications like Bazaar [1], Internet Relay Chat etc. where structure, reference and citation structure of student contributions is lost. Consequently, these representations suffer from sequential incoherence [13].
2. Threaded:
   Examples include *Hermes* [15] and MOOC discussions. Characteristics include the protracted pace of collaboration, and explicit post-reply/citation structure.
3. Graph based:
   This is very common among argumentation systems like *Belvedere* [9], Reason!Able [10] etc. which can be adapted to a collaborative domain, as well as community knowledge building tools such as Knowledge Forum [11]. It can be thought of like a citation network where nodes indicate concepts/ideas and edges indicate build-upon relations, support for/against it etc.
4. Container Based:
   One example of this is SenseMaker [12] which organizes ideas/propositions into frames which can be nested if required. It differs from graph based representations in that relations between propositions are implicit. Nesting implies support and opposition between two propositions is not directly indicated. This representation is commonly used in debate style collaboration systems where there are a limited number of opinions collaborators can have, and the 'for/against' relationship between them is obvious.

In addition to supporting any of the above UIs, some tools such as *Auracaria [14]* also support convertibility between multiple UI representations.

The varied nature of possible UIs and the need for convertibility between them have three implications:

1. BOTH the view and the controller MUST be customizable. The XBlock must not restrict these in any way. Responsibility for providing the View and Controller using HTML+CSS and JavaScript respectively, lies with the course creator using the XBlock.
2. The XBlock must implement a Model on the server side with an API that is agnostic to the type of collaborative activity/UI involved.
3. The XBlock must allow as much customizability via EdX studio without needing code changes on production servers.

The second aspect of the UI involves deciding the actions that students can perform with it. For example, in a synchronous chat setting, students can be restricted to posting messages with certain sentence openers by pressing specific buttons to post a message. Similarly, scaffolding of individual student activities can be provided at the UI level itself as well. Alternatively, in a scenario like knowledge forum, there may be a ‘toolbox’ panel for students to add new notes, add attachments etc. Since the View and Controller components supplied by the course creator completely specify the look and functionality, such customizations do not affect the XBlock design as long as the Controller translates student actions via the UI (View) to the Model using the API exposed by the Model.

Note:

Since the Model stores information about the collaborative interaction and the Model is independent of the View, the Controller is responsible for handling synchronous vs. asynchronous behavior as well. If the Controller keeps fetching updates from the Model at short intervals and updates the View accordingly, it simulates a synchronous experience. Alternatively, if it only fetches updates on page load, it simulates an asynchronous experience.

Having established this understanding of functionality to be offered by the XBlock, the following sections will demonstrate a pipeline that will be followed by the XBlock. The legend to interpret the diagrams has been provided below as well.

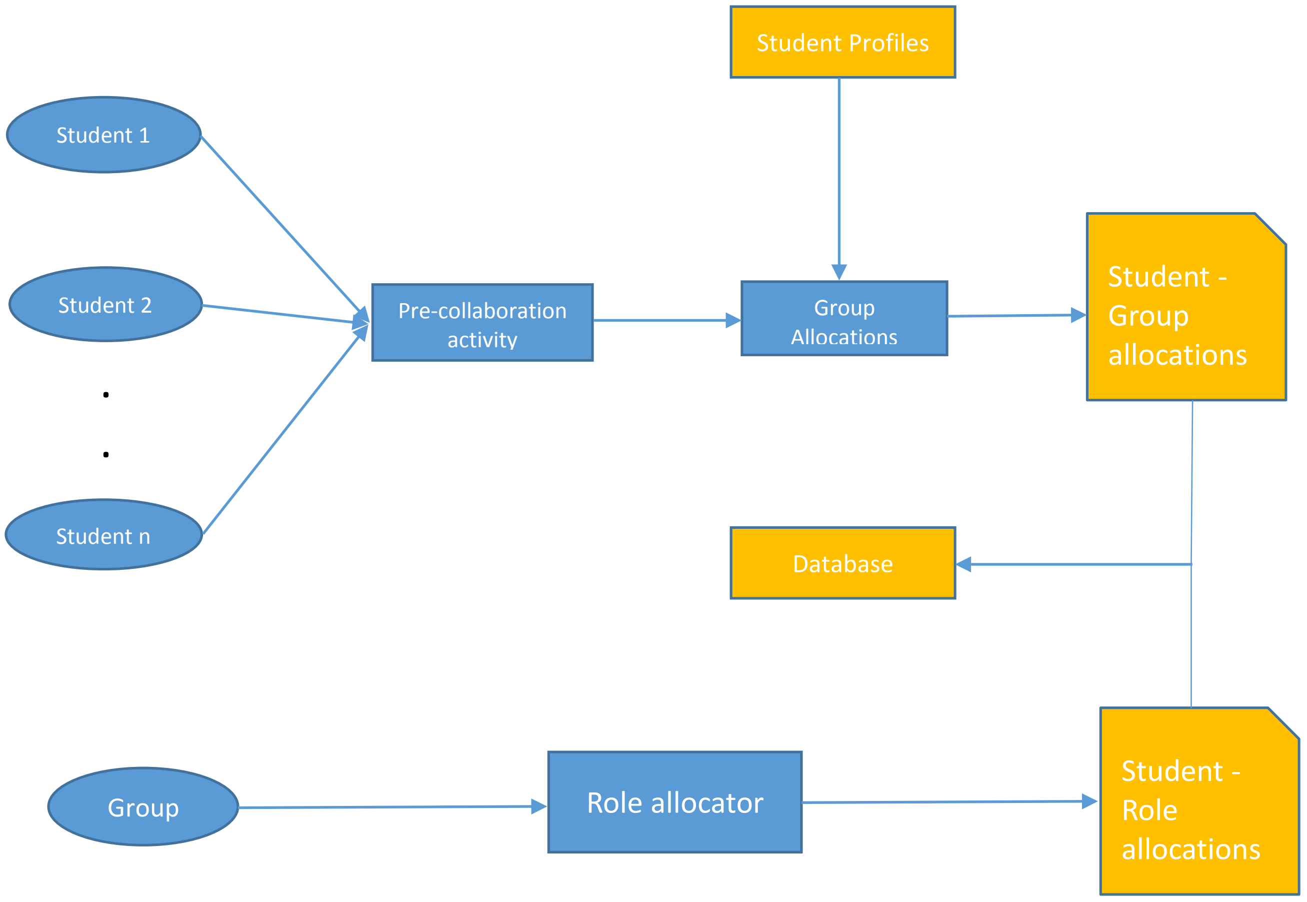
⬭ : Student/participant(s)

🟦 : Pipeline Process

🟨 : Data store/ Consumable data

→ : Student actions/Data Flow/Process flow

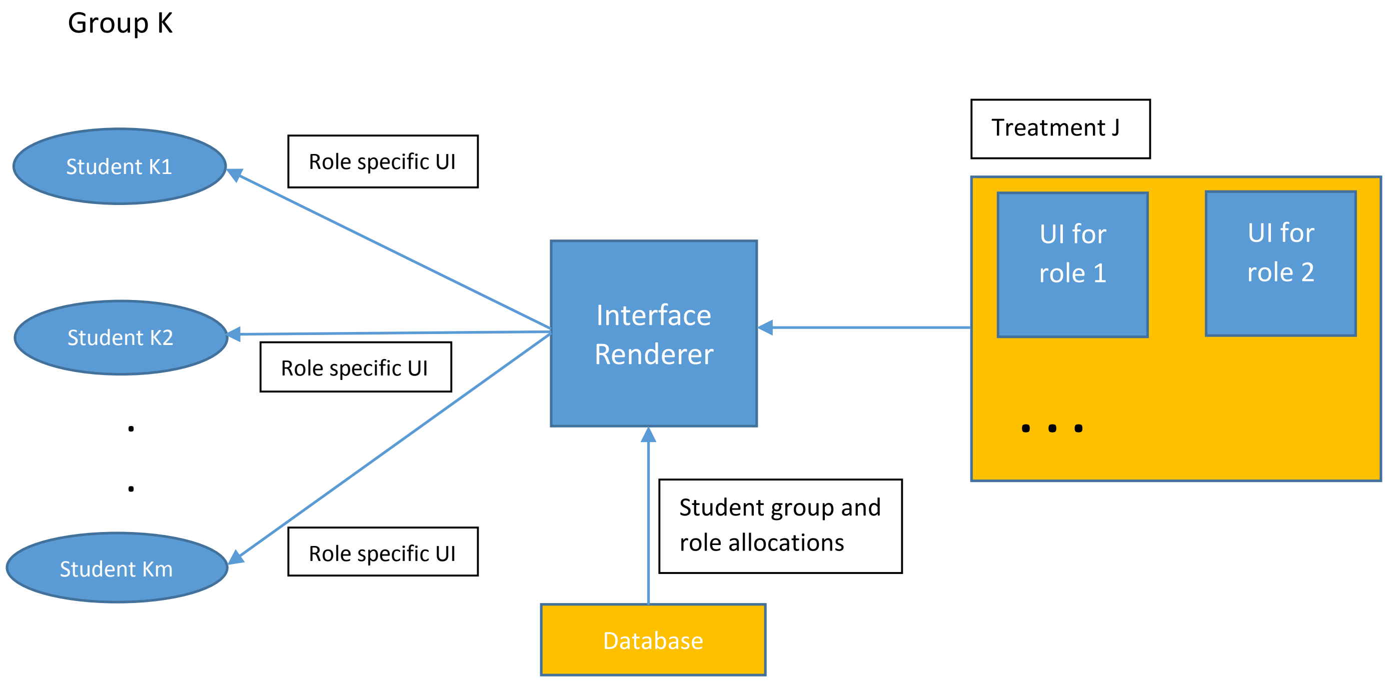
Pre-collaboration and group formation:



The pre-collaboration step is an individual activity which can be used to make the collaboration script explicit to the student or to provide preparation materials. This can be in the form of text, audio or video. All three can be specified using EdX studio without having to make code changes. The group matching step is used to form sub-sets of students who will then engage in a collaborative activity on a group level. The Role Allocation step involves assigning students roles that have specific responsibilities in the collaborative activity as dictated by the script designed by the course instructor. By default, both matching and role allocation will be random. Any other group formation algorithms with will have to be implemented in code by the course creator.

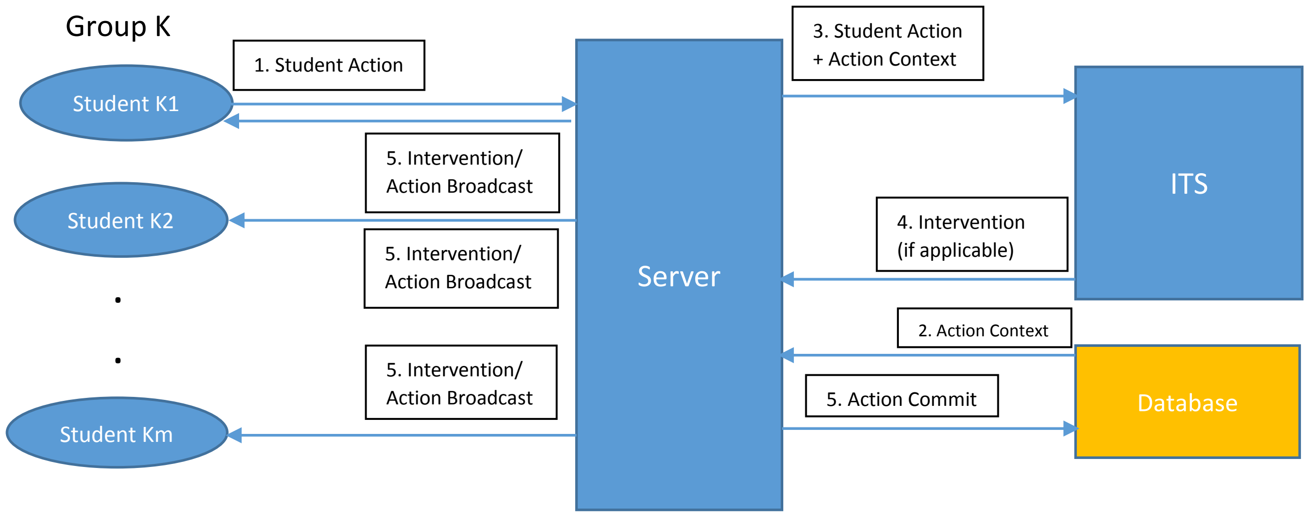
For purposes of experimentation, the order of these three steps can be configured and steps can be left out if required. If groups/roles have been assigned before the pre-collaboration, pre-collaboration materials can be targeted based on those parameters.

UI Rendering:



The interface renderer is a part of the XBlock that is responsible for delivering the UI to the student. To specify a UI, HTML and CSS must be provided. To specify the functionality of the UI, JavaScript must be provided as well. A 'Treatment' is a set of role specific UIs that can be allocated to students within a group. Treatments can be used to provide different UIs to students performing the same role in different groups, for experimental purposes. A distribution of treatments across groups will have to be specified. Treatments and their distributions over groups can be specified via EdX studio without having to make any code changes.

Intelligent Tutoring System (ITS):


Group K
Student K1
Student K2
Student Km
1. Student Action
5. Intervention/ Action Broadcast
5. Intervention/ Action Broadcast
5. Intervention/ Action Broadcast
Server
3. Student Action + Action Context
ITS
4. Intervention (if applicable)
2. Action Context
5. Action Commit
Database

Due to the highly varied and customizable nature of Intelligent Tutoring Systems, their functionality will not be built in to the XBlock. There are two options to add this functionality:

1. ITS hosted as an external service
2. Modifying the XBlock implementation

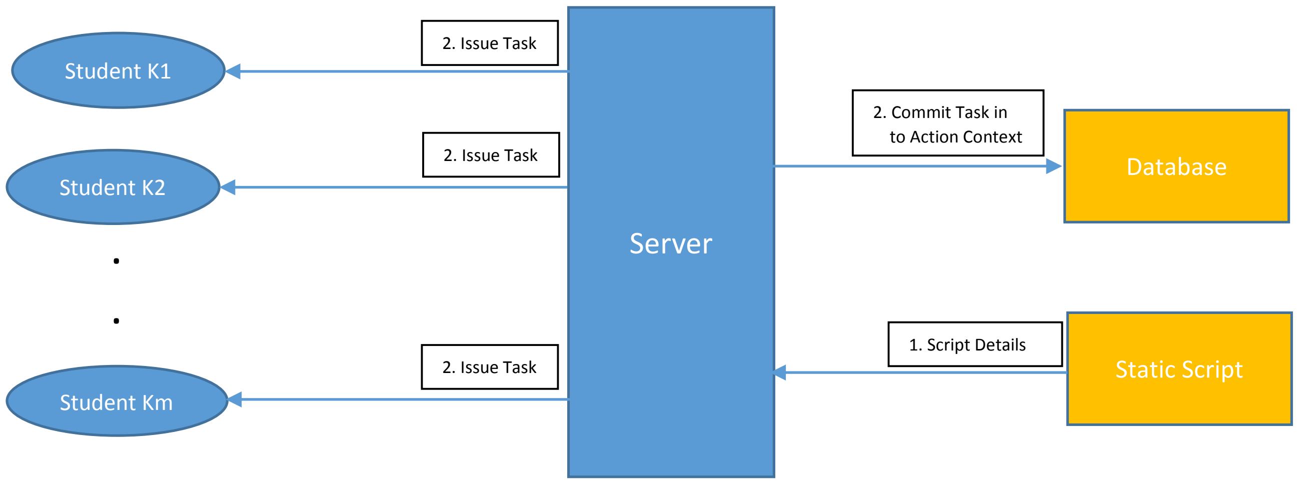
The external service method is ideal as it ensures modularity of components, will not need a code review before use in EdX courses, can be configured from the EdX studio and does not require any knowledge of the XBlock code on the part of the course creator using the XBlock. However, there should be an agreement over what features/action context is passed to the service. Additionally, EdX may place certain restrictions on what information can be sent to the external ITS service. If the service is deployed within the EdX environment, those restrictions shouldn't apply.

The second method involves modifying the XBlock code itself to achieve the desired functionality. This would require some knowledge of the XBlock code. Moreover, in order to use the XBlock on production EdX servers, a code review would be necessary. However, potentially complex ITS behaviors can be achieved, using any necessary data about the collaborative activity.

Note: In the scenario depicted in the above figure, handling the intervention at the UI level is the responsibility of the Controller.

Static Script:

Simple automated behaviors such as static scripts can be supported natively by the XBlock in the absence of an ITS using a similar design as above. In addition, the Server may have to issue tasks based on the static script in which case the following would occur:



The static script could be provided in the form of text via EdX studio. The expected capabilities of the static script and a format in which to specify it will have to be agreed upon.

Useful Resources:

To learn more about the XBlock framework, visit the following links:

http://edx-installing-configuring-and-running.readthedocs.org/en/latest/devstack/install_devstack.html

http://xblock.readthedocs.org/en/latest/

http://opencraft.com/doc/edx/xblock/tutorial.html

Also visit http://dance.cs.cmu.edu/resources/ for a listing of various community contributed XBlocks.

Citations:

[1] Scardamalia, M., & Bereiter, C. (1994). Computer support for knowledge-building communities. Journal of the Learning Sciences, 3(3), 265-283.

[2] Gokhale, Anuradha A. "Collaborative learning enhances critical thinking." (1995).

[3] Weinberger, Armin, Bernhard Ertl, Frank Fischer, and Heinz Mandl. "Epistemic and social scripts in computer–supported collaborative learning." *Instructional Science* 33, no. 1 (2005): 1-30.

[4] Ke, Fengfeng, and Kui Xie. "Toward deep learning for adult students in online courses." *The Internet and Higher Education* 12, no. 3 (2009): 136-145.

[5] Keeton, M. T. “Best online instructional practices: report of phase of an ongoing study.” Journal of Asynchronous Learning Networks, 8(2) (2004), 75−100.

[6] https://novoed.com/venture1-2014-2#

[7] http://www.bloomberg.com/bw/articles/2014-02-11/behold-a-virtual-course-without-online-eds-huge-dropout-rate

[8] Adamson, David, and Carolyn Penstein Rosé. "Coordinating multi-dimensional support in collaborative conversational agents." In *Intelligent Tutoring Systems*, pp. 346-351. Springer Berlin Heidelberg, 2012.

[9] Suthers, D. D., Connelly, J., Lesgold, A., Paolucci, M., Toth, E. E., Toth, J., et al. (2001). Representational
and advisory guidance for students learning scientific inquiry. In K. D. Forbus & P. J. Feltovich (Eds.),
Smart machines in education: The coming revolution in educational technology (pp. 7–35). Menlo Park:
AAAI/MIT.

[10] Van Gelder, Tim. "Argument mapping with reason! able." *The American Philosophical Association Newsletter on Philosophy and Computers* 2, no. 1 (2002): 85-90.

[11] Scardamalia, Marlene. "CSILE/Knowledge forum®." *Education and technology: An encyclopedia (2004): 183-192.*

[12] Bell, Philip. "Using argument representations to make thinking visible for individuals and groups." In *Proceedings of the 2nd international conference on Computer support for collaborative learning*, pp. 10-19. International Society of the Learning Sciences, 1997

[13] McAlister, S., Ravenscroft, A., & Scanlon, E. (2004). Combining interaction and context design to support collaborative argumentation using a tool for synchronous CMC. Journal of Computer Assisted Learning: Special Issue: Developing Dialogue for Learning, 20(3), 194–204.
[14] Reed, Chris, and Glenn Rowe. "Araucaria: Software for argument analysis, diagramming and representation." *International Journal on Artificial Intelligence Tools* 13, no. 04 (2004): 961-979.

[15] Karacapilidis, N., & Papadias, D. (2001). Computer supported argumentation and collaborative decision making: The Hermes system. Information Systems, 26(4), 259–277.