CS6460 Educational Technology
Project Proposal: **Task Authoring Tool for Computer-Based Testing**
Track: Development
Matthew Hull

I. Introduction:
This proposal details a plan for developing a tool that captures the modeling of learning objectives used for instructional design. Learning objectives are described as Tasks, something that a learner can *do* at the completion of learning.

This tool allows a courseware/instructional designer to input Tasks electronically and then use them for creating a computer-based test.

II. Problem and Motivation:
Within the discipline of instructional systems design (ISD), there exists a lack of technologies that allow task analysis and modeling. Task modeling is a critical component in the instructional design cycle where learning objectives are identified prior to the development of content. Current course design tools are heavily focused on content development and do not offer this capability. When instructional designers and subject matter experts (SMEs) build courses without a task analysis, they are often faced with several problems:

1. An extensive amount of work in reverse engineering the learning objectives after the course is built.

2. Finding that the course that has been developed does not actually meet their original intent.

3. Difficulty in analyzing learning outcomes without structured data

4. Difficulty in designing tests that measure student ability against learning objectives.

The motivation behind developing this tool is to primarily focus on the stated problem 4. (above): Modeling a task and designing computer-based tests on the resulting task analysis. A test built on a task analysis solves the stated problem 3.: It allows data analysis to be performed on structured data and problem 2.: The designers can state the learning objectives *vis-a-vis* the promulgation of tasks from the beginning of the course design process. Finally, the stated problem 1. is solved: A task analysis tool would alleviate the need to reverse engineer learning objectives.

III. Existing Solutions:
The existing solutions for task modeling and task analysis are sparse. The primary solution is to accomplish a task analysis as an academic exercise. Several publications detail how to accomplish task analysis but do not offer any tools, i.e., applications for it. In the best case, a published method for Task Analysis offers a

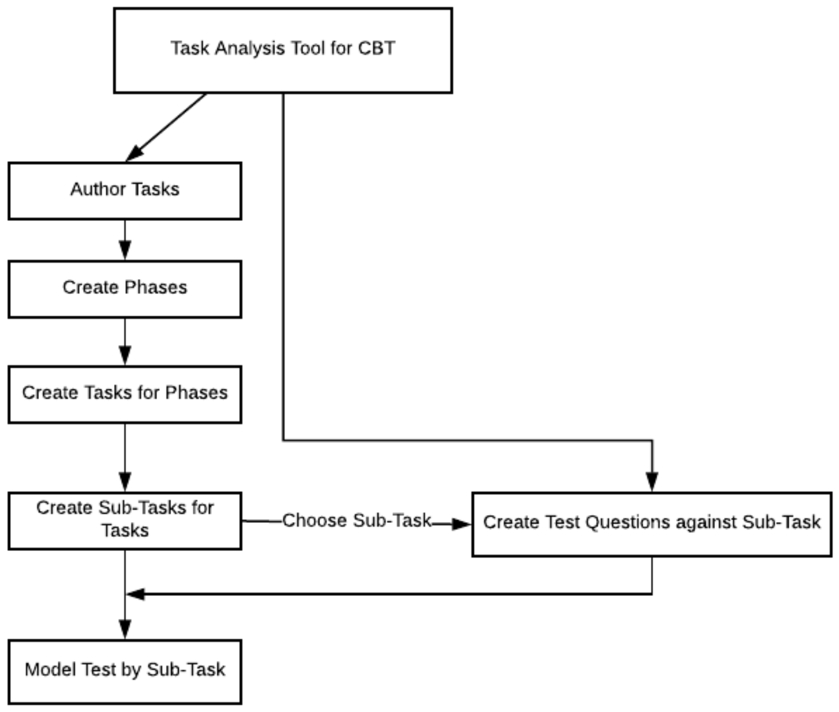
template for recording a written description of a task. Learning Management Systems (LMS) such as Moodle offer a means to categorize quiz/test questions according to categories and subcategories; however, this categorization generally does not occur until after the course development has taken place.

There is a clear need for a method to electronically persist a task analysis/modeling effort. The tool proposed would allow someone to design a course using a top-down approach: author learning objectives in the form of tasks, i.e., a task describes what the course designer/SME wants the learner to be able to do at the completion of learning.

The existing solutions above do not offer a way to transfer the Task Analysis into anything meaningful that can be used for other parts of learning design. i.e., it is very tedious to take a written task analysis and then author test questions based on the various tasks. This proposal describes a tool to author the task electronically and use it as a category for testing. Further details on this will be provided in the Design section of this proposal.

IV. Design:
This tool will be designed as a graphical user interface to a web application that allows creation of tasks for a course of study in a hierarchical manner. The tasks will be saved to a database and with based create, read, update, and delete (CRUD) functions.

Second to task authoring is creating test-questions *against* a task model. The tool will allow input of test questions that can be written and tagged with the appropriate sub-task. The following diagram illustrates the main functions:


Task Analysis Tool for CBT
Author Tasks
Create Phases
Create Tasks for Phases
Create Sub-Tasks for Tasks
Choose Sub-Task
Create Test Questions against Sub-Task
Model Test by Sub-Task

IV. a. Introduction to 3-Level Task Analysis

This tool enables the task description method provided in Task Analysis Method's for Instructional Design (Jonassen, et., al. 1999). Tasks are described hierarchically in three levels. This tool adopts three levels of task analysis, referred to as **Phases**, **Tasks**, and **Sub-Tasks**. Each level is defined and named below.

1. The first level is major tasks of learning that serve as the milestones.

   e.g., Two major tasks for a Data Analysis course might be:

   1. *Apply Data Pre-Processing Methods*
   2. *Perform Logistic Regression*

These major tasks are henceforth called **Phases**.

2. The second level is tasks that describe the components of the major task.

   e.g., The tasks that describe Applying Data Pre-Processing Methods (above) could be:

1. *Apply Data Pre-Processing Methods*
   - *1.1 Processing Missing Data*
   - *1.2 Process Outliers*
   - *1.3 Substantiate Data Types*

These tasks are simply called **Tasks**.

3. The third level in the hierarchy are sub-tasks that describe even more specifically what the task is.

   e.g., Keeping with the example above, sub-tasks are provided for task 1.1 Processing Missing Data.

1. *Apply Data Pre-Processing Methods*
   - *1.1 Processing Missing Data*
     - *1.1.1 Identify Missing Data*
     - *1.1.2 Substitute Missing Data*

These are referred to as **Sub-Tasks**.

IV. b. Introduction to Test Question Authoring Against a Task Analysis

As highlighted in the motivation above, the tasks designed by the SME can be transferred into a concrete measurement of learning objectives through testing. The tool will be designed to allow input of test questions that are tagged against with a task. The outcome of this is twofold: It guides the writing of test questions to measure only the intended learning objectives and structures the data such that a test can be created and analyzed efficiently.

V. Technical Specifications
The tool will make use of a web application that is implemented using the following technology stack:

Front-end: HTML, Javascript / jQuery
Server-side: PHP, Custom written classes
Database: MySQL

In this project, a user will input a task using a graphical user interface to a web application. The web application makes use of HTML/Javascript to send information about a task to the server. PHP will be used to persist and retrieve the input information to a MySQL database.

VI. Integrations and External Resources
This tool will integrate with an existing tool that I have already developed for computer based-testing. The existing tool allows input of questions and generation of tests using *static* categories. The integration with the existing application adds the capability of tasks authoring for use with CBT.

VII. Task List & Calendar

For practicality, the task list, hours required and project calendar are integrated here with milestones and deliverables. For each week, a list of tasks and sub-tasks that must be accomplished are provided. Each task has a designated amount of hours for estimated completion time. Each week contains a total amount of planned hours for each week. A running total of hours, termed 'To-Date Hours' is provided to show the total work time elapsed for each week of the project. At the completion of each week, a milestone or deliverable is provided. For some weeks a deliverable could be a submission of documents, prototypes or presentation. On other weeks where programming is being performed, a status update will be communicated to the Mentor.

Week 8

Total Time: 12 Hours
To-Date Hours: 12 Hours

1. Write Use Cases / User Stories - Documentation - 2 Hours
    1.1. Brainstorm User Stories
    1.2. Use Lucidchart to Produce User Stories Diagram

2. Propose technical classes describing task analysis tool - Documentation - 2 Hours
    2.1. Brainstorm Initial Classes
    2.2. Write Class Relationships
    2.3. Use Lucidchart to Produce Class Diagram

3. Propose Application Programming Interface between Client and Server-Side - 6
    3.1. Determine Information Produced by Server-Side Classes
    3.2. Designate Information Produced by Client-Side
    3.3. Produce API Documentation Using Markdown or Lucidchart

4. Write Software Tests - Documentation - 2 Hours
    4.1. Write User Test Cases against User Stories
    4.2. Author Ad-Hoc Test Cases as Needed to Cover Additional Functionality
    4.3. Use Markdown Editor to Produce Testing Plan Document

End of Week 8 Milestone/Deliverable:
All design documents from Tasks 1-4

Week 9
Total Time: 10 Hours
To-Date Hours: 22

5. Develop Prototypes - Documentation - 10 Hours
   5.1. Sketch Wireframe
   5.2. Develop Prototype Interaction Story
   5.3. Use Lucidchart to Produce Prototype

<u>End of Week 9 Milestone/Deliverable:</u>
All prototype documents for Task 5

Week 10
Total Time: 12 Hours
To-Date Hours: 34

6. Present Intermediate Milestone 1 - Low Fidelity Prototype - 6 Hours
   6.1. Author Presentation
   6.2. Produce Video / Narration as Needed
   6.3. Invite Classmates/Mentor to Test Prototype
   6.4. Incorporate Feedback Into Project Plan as Needed

7. Develop Classes - Programming - 6 Hours
   7.1. Consult Class Diagram from Task 1
   7.2. Develop Classes in Using PHP Language

<u>End of Week 10 Milestone/Deliverable:</u>
Intermediate Milestone 1 Presentation
Invitations to Beta Test Prototype

Weeks 11
Total Time: 14 Hours
To-Date Hours: 48

8. Populate Database Tables with Classes - 10 Hours
   8.1. Use PHP MyAdmin Tool to Edit MySQL Database
   8.2. Input Classes according to Task 1
   8.3. Develop Sample MySQL Queries to Test Table Creation

<u>End of Week 11 Milestone/Deliverable:</u>
Status Update with Mentor - Communicate Challenges Encountered and Needed Revisions

Week 12
Total Time: 20 Hours
To-Date Hours: 68

9. Develop UI Elements - Programming - 12 Hours
    9.1. Consult Existing User Stories Task
    9.2. Consult Existing Prototype Task
    9.3. Develop UI Elements using HTML
    9.4. Develop UI Functionality using Javascript/jQuery

End of Week 12 Milestone/Deliverable:
Status Update with Mentor - Communicate Challenges Encountered and Needed Revisions

Week 13
Total Hours: 16
To-Date Hours: 84

10. Develop Javascript - Server Interface - Programming - 8 Hours
    10.1. Consult Existing API Task
    10.2. Develop API Interface Using Javascript
    10.3. Develop PHP Server Scripting - Programming

11. Develop UI Elements - Programming - 8 Hours
    11.1. Consult Existing Prototype
    11.2. Consult Existing User-Stories
    11.3. Style Elements as Required Using CSS

End of Week 13 Milestone/Deliverable:
Status Update with Mentor - Communicate Challenges Encountered and Needed Revisions

Week 14
Total Hours: 10
To-Date Hours: 94

12. Execute Testing Plan - Testing - 6 Hours
    1. Consult Existing Testing Plan Task
    2. Execute Testing
    3. Make Modifications as Required
    4. Write New Tests as Needed

13. Present Intermediate Milestone 2 - Functional Prototype - 4 Hours
    13.1. Author Presentation
    13.2. Produce Video / Narration as Needed
    13.3. Invite Classmates/Mentor to Test Prototype if Possible
    13.4. Incorporate Feedback into Project Plan

End of Week 14 Milestone/Deliverable:
Presentation of Milestone 2 Functional Prototype
Invitations to Test and Evaluate the Prototype

Week 15
Total Hours: 8
To-Date Hours: 102

14. Conduct Task Analysis Using Tool on CSE 6242 - Data Visualization and Analytics - 4 Hours
    14.1. Consult Curriculum
    14.2. Develop Task Analysis Using Jonassen et. al. Methodology
    14.3. Consult Fellow CSE 6242 TAs for Feedback
    14.4. Input Tasks into Tool

15. Author Sample Test Questions on CSE 6242 - 4 Hours
    15.1. Using Tool, Develop Sample Questions
    15.2. Tag Questions with Appropriate Task

End of Week 15 Milestone/Deliverable:

Week 16
Total Hours: 10
To Date Hours: 112

16. Author Final Paper - 5 Hours
17. Author Presentation - 5 Hours

End of Week 16 Milestone/Deliverable:
Presentation / Paper

References:

Moodle - Open Source Learning Platform. (2018). Retrieved January 28, 2018 from https://moodle.org

David H. Jonassen, Martin Tessmer, and Wallace H. Hannum. (1999). Task Analysis Methods for Instructional Design. Routledge.