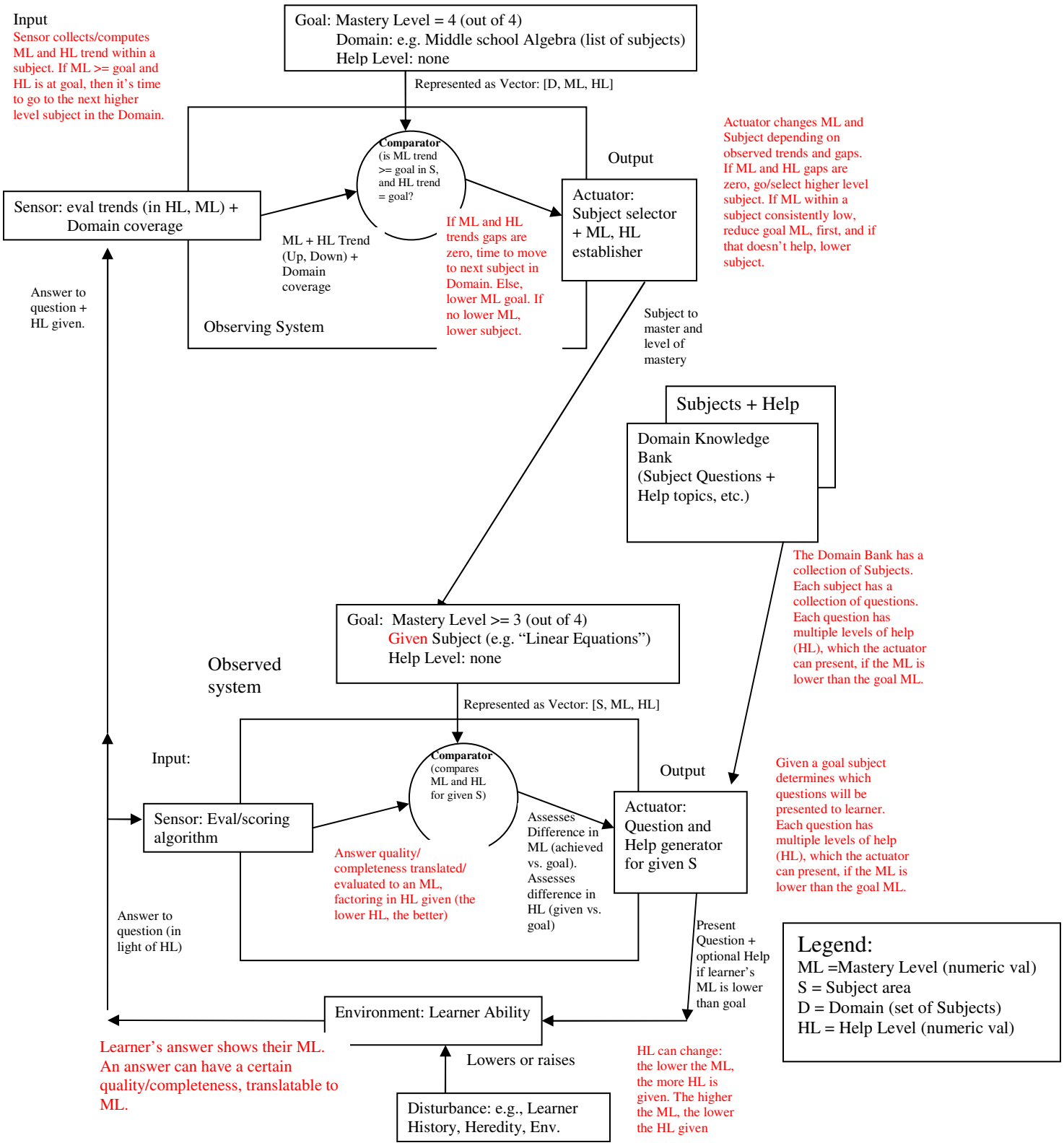
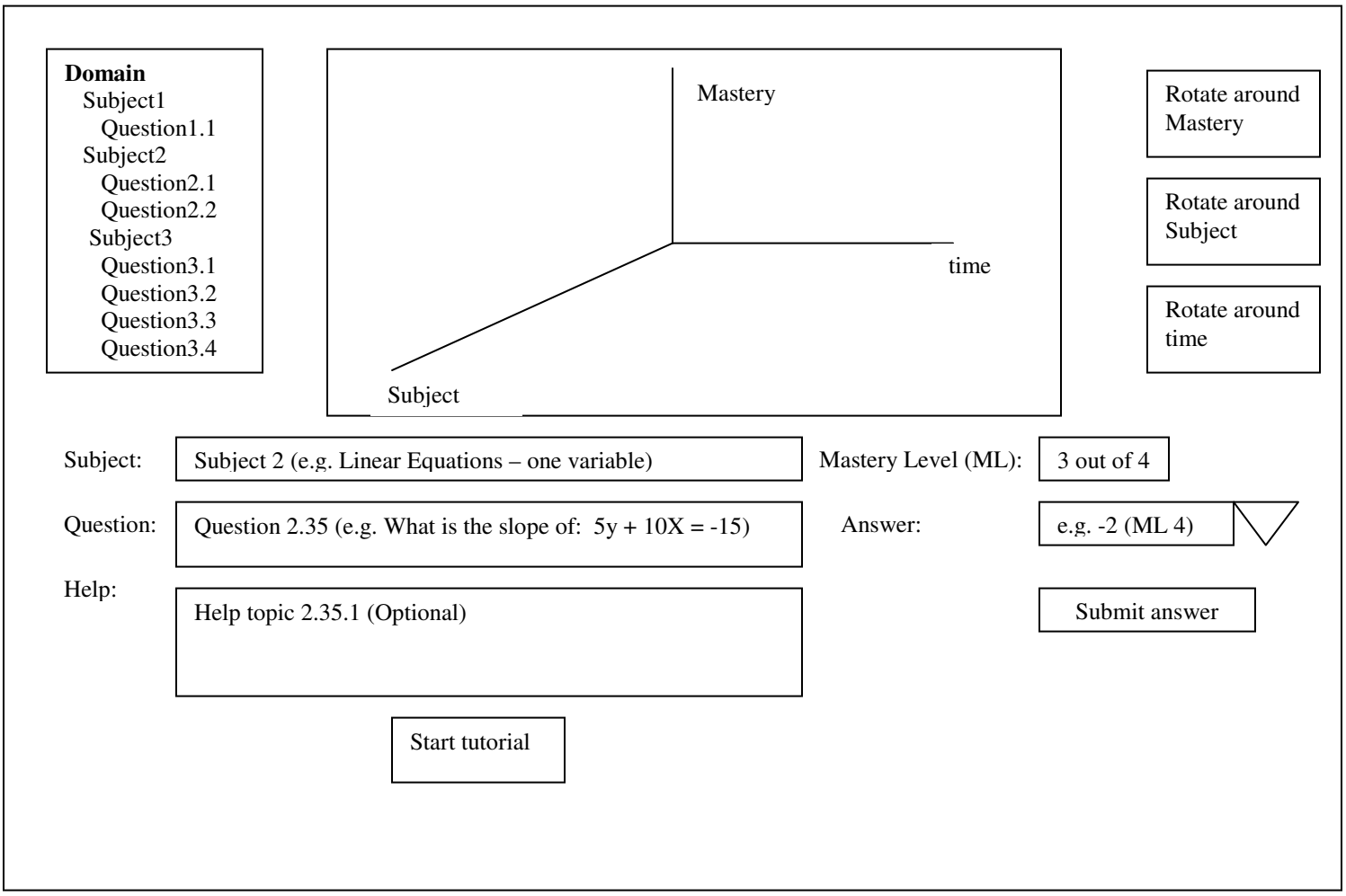
Final Project – Second Order automatic Feedback System 11/16/07

Automatic Tutoring System



GUI Spec:



Use cases/scenarios

1. Initialization
    a. The learner starts the tutorial by pressing the "Start tutorial" button
    b. The system picks a subject from the domain to be mastered and displays it in the "Subject" field (read-only)
    c. The system picks a question from the selected subject and displays it in the "Question" field (read-only)
        i. Each question has a system-assigned ML associated with it, based on its difficulty
    d. No help topic is initially given in the "Help" field (read-only)
    e. No Mastery Level (ML) is displayed in the "Mastery Level" field (read-only)
        i. The system expects a ML of at least 3 out of 4 and will teach to that goal
    f. The system is waiting for the learner's response
2. Learner selects the answer that they think is correct from the drop-down list labeled "Answer"
3. Learner hits the "Submit answer" button
4. The system "calculates" the "Mastery Level" (ML) value associated with the learner's answer and compares it to the goal ML (set by the Tutoring System)
    a. If the answer ML >= goal ML, the system will pick another question at the goal ML difficulty
        i. Once the questions for the current subject are exhausted, and provided the learner is still performing well (student ML >= goal ML), the system switches to the next Subject within the domain
    b. If answer ML < goal ML (learner underperforming), help topics will be displayed to aid the learner
        i. More detailed help is displayed in response to lower learner ML.

ii. When the help topics for that particular question are exhausted, the system will switch down to a prerequisite subject (since it assumes the learner is struggling with prior/prerequisite concepts)

Data model:

Domain – a list of Subjects
- The Domain has a designated first Subject (to help select a Subject appropriate to student's ML)
- each Subject is linked to "supporting" subjects (prerequisites)
- each Subject is linked to Subjects "supported" by it (progression path to next Subject)
- The Domain has a count of Subjects within it

Note: within the domain, there are only dependencies ("supports" and "supporting") between Subjects, but not within a Subject (between the Questions). If someone needs dependencies among Questions, they should be separate the Qs into two different Subjects, depending on each other. So this scheme of one level dependence is flexible enough.

Subject – a list of Questions
- each Subject has a count of Questions within it
- has "completed" field to indicate student answered all Qs within Subject
- has a Subject-ML to capture learner ML for this Subject

Question – submitted by the system; requires student response (Answer)
- has links to Equivalent Questions (Questions about the same material, at the same level of difficulty)
  - each Question has Number of Equivalent Questions
- has a link to Help topics relevant to this Q. Each Question has Max HL help topics
- has a link to all possible Answers for this Q (answer options, like multiple choice)
- has "answered" field to indicate it has been displayed and answered by student

Answer – a system-displayed answer, picked by learner
- has a score associated with it, mapped to the ML (e.g. fully correct -> 4, almost correct -> 3, etc.)

Help – a system-displayed help topic, available when student is underperforming
- additional topics (more detailed help) are displayed if learner's ML drops

The Domain Model

- A multi-dimensional representation of the subject knowledge (topics, relationships, etc.) overlaid with a learner's profile (mastery, interest, etc.)
-

Pseudo Code:

```
[SOL-vector] = initialize-SOL()
[FOL-vector] = initialize-FOL()
while (learner-ML <= final-ML-goal and more-subjects)
        FOL:
                FOL-ML-performance = comparator(FOL-ML-goal, learner-ML)        // measure learner performance gap
                [question, help] = FOL-actuator(FOL-ML-performance, subject)    // determine what to display to learner
                answer = display-to-learner(question, help)                     // learner action/response
                learner-ML = FOL-sensor(answer, question)                       // assess/eval the answer
        SOL:
                // same as FOL: learner-ML = SOL-sensor(answer)                 // SOL and FOL have same sensor, input, output
                SOL-ML-performance = comparator(Final-ML-goal, learner-ML)      // SOL and FOL have same comparator function, different inputs
                [FOL-ML-goal, subject] = SOL-actuator(SOL-ML-performance)
end-while
```

```
FOL-actuator(FOL-ML-performance, subject)
        If (FOL-ML-performance >= 0)        // learner doing OK
                question-type = new
                question = select-question(subject, question-type) // round robin on new questions within the given subject. Mark each Q as displayed
                help = null             // no help topic needed
                HL = 0                  // flag
        Else                            // learner underperforming
                question-type = equivalent                      // similar to the one learner underperformed on
                question = select-question(subject, question-type)// round robin on equivalent questions for failing question
                If (HL < max-HL) HL = HL + 1                    // increment up to max help level supported by system
                help = build-help(HL, question)                 // concatenate 1-to-max help topics together for the selected question
        end-if
        return(question, help)
end

FOL-sensor(answer, question)
        Return(Learner-ML = calculate-student-score(answer))            // answer.score field
end

SOL-actuator(SOL-ML-performance)
        ML-trend = calculate-trends(SOL-ML-performance)
        If (SOL-ML-performance >= 0)                    // learner doing OK
                If (subject marked "completed")         // completed current subject
                        subject.subject-ML = learner-ML // capture ML for this subject, before moving to next
                        subject-type = supported-subject // next, higher-level subject
                        subject = select-subject(subject, subject-type)
                        initialize-new-subject(subject) // mark "incomplete", etc.
                        FOL-ML-goal = final-ML-goal     // raise the bar to desired goal
                        lowered-ML = false              // start with high goal again
                        reset-trends()
                end-if                          // if current subject not completed, stay with same subject
        else                                    // learner underperforming
                If (ML-trend < acceptable)
                        If (lowered-ML)     // already lowered ML once
                                subject-type = supporting-subject       // need to drop to a prerequisite subject
                                subject = select-subject(subject, subject-type)
                                initialize-new-subject(subject)         // mark "incomplete", reset trend levels
                        else                // give learner a chance at a lower ML-goal level
                                lower-ML = true
                                FOL-ML-goal = final-ML-goal – 1         // lower the bar once below desired goal
        end
        return(FOL-ML-goal, subject)
end
```

```
initialize-SOL()
        domain = read-domain()                  // subjects, questions, help topics, expected answers
        for each subject within domain:
                initialize-new-subject(subject) // mark "incomplete", reset trend levels
                                                // each question within each subject marked "not answered"
        final-ML-goal = 4
        max-HL = 2                              // number of help levels per question
```

```
        learner-ML = final-ML-goal                      // assume learner will perform well
        subject = select-subject(domain, learner-ML)    // select initial subject, based on assumed learner ML
        more-subjects = true                            // flag to indicate completion of domain (if no more subjects within it)
end

initialize-FOL()
        FOL-ML-goal = final-ML-goal         // start at desired goal level
        HL = 0                              // flag indicating actual help level
        question-type = new                 // start with new question
end

initialize-new-subject(subject)
        subject.state = incomplete
        HL = 0
        for each question within subject
                question.state = not-answered
end

reset-trends()
        ML-trend = 4                // high/goal-ML is good
        HL-trend = 0                // no/low help is good
end

select-subject(subject, subject-type)
        // select either supporting or supported subject
        // need to deal with boundary conditions: no more supported or supporting subjects
end

select-question(subject, question-type)
        if (question-type == new)
                select next question in round robin fashion
        else                    // need equivalent question
                select equivalent question to current question
        end
        mark question as “answered”
        If (all answers market “answered”)      // don’t worry about marking answers of equivalent questions
                mark subject “completed”
end

comparator(ML-goal, learner-ML)
        return (gap = learner-ML - ML-goal)  // if negative, learner performs poorly; >= 0 is good
end
```

Design Questions/Tradeoffs:

- Should the FOL be designed as an independent system, that can function in some intelligible way even if disconnected from the SOL?
- Should the SOL "aspire" to pull the student to a higher ML

Possible To-Dos/extensions

- on the knowledge-map (k-map; tree browser for now)
    - learner can explore the domain by selecting a subject, asking what it is about, and starting to work in that area of the domain (e.g., question/answer sessions)
    - learner can expand the domain by adding topics to the knowledge-map (after demonstrating mastery, or gaining authority in some other way)
        - Implies a k-map (or at least subject/topic) Editor to allow putting in content, in a standard or templated format.
- on the k-map or the domain 3D cube
    - the system will capture and display (pop-up, mouse-over, etc.) time spent exploring the subject
        - this potentially indicates learner's level of interest
    - learner can select a subject and enter in the system
        - notes, assessment, level of interest (subjective evaluation)
- Learner (and teacher, SME, others) should be able to load other people's annotated, expanded k-maps
    - Compare similarities, overlapping interests, etc.
    - Identify differences/gaps (in mastery, interests, etc.)
    - This obviously implies being able to upload/download/exchange/share k-maps with others
        - Could develop a marketplace for k-maps
            - Communities of "birds of a feather"
            - Tutor-learner relationships/opportunities
            - Domain/knowledge SME/supplier – consumer relationships/opportunities
            - Implies mappable/equivalent/standard vocabularies, structures, etc. (challenge)
                - Use of topic maps? (XTM, http://en.wikipedia.org/wiki/Topic_Maps)
- Learner should be able to ask the system for suggested navigation paths through the domain based on
    - Their history of mastery (learning style, learning ease/difficulty, etc.)
    - Their indication of interests
        - Derived from experience within the domain (interest in pre-req subjects would lead to suggestion of post-req subjects), or
        - Experience in other, related domains/k-maps
            - This implies linking across domains which is challenging
                - Requires use of mappable/equivalent/standard vocabularies, structures, etc. (XTM, http://en.wikipedia.org/wiki/Topic_Maps)
    - Someone else's (SME, teacher, other learner) recommendation

Learning Theory

- Learner is an active constructor, not a passive recipient of knowledge (A. Brown, UCB – Educational Researcher, 11/94)
- Communities of Learners are effective: "reciprocal teaching involved the development of a mini-learning community, intent not only on understanding and interpreting texts as given, but also on establishing an interpretive community (Fish, 1980) whose interaction with texts was as much a matter of community understanding and shared experience as it was strictly textual interpretation. (A. Brown, UCB – Educational Researcher, 11/94)
- A zone of proximal development defines the distance between a performer's current level of learning and the level s/he can reach with the help of people, tools, and powerful artifacts (Vygotsky, 1978)
- We are better able to design a spiraling curriculum such as that intended by Bruner (1969 – On Knowing: essays for the left hand)