15.01.2020

# Словарь (Глоссарий)

|  |  |
| --- | --- |
| Упражнение (задача, вопрос) |  |
| Ученик (студент) |  |
| Учитель (преподаватель) |  |
| Предметная область (Пр.обл.) |  |
| Закономерность | То в предметной области, что должен знать (понимать) студент |
| Понимание закономерности студентом |  |
| Вероятность понимания закономерности |  |
| Вопрос | = Упражнение |
| Тип вопроса | Вид постановки условия и форма запрашиваемого ответа, например:  - (множественный) выбор  - соответствие  - натаскивание (drag and drop)  - открытый ответ:  - число  - текст (строка) |
| Ошибка |  |
| Объяснение |  |
|  |  |
| Логическое ядро | (backend) |
| Вопросная система |  |
| Клиентский интерфейс  (Интерфейс конечного пользователя) | Реализует взаимодействие с конечными пользователями через собственный интерфейс (frontend).  Преобразует вопрос из универсального представления Вопросно-ответной системы в подходящую (родную для себя) форму.  Даёт обратную связь студенту (объяснение ошибок, возможно личная статистика).  Сообщает Вопросно-ответной системе типы поддерживаемых вопросов.  Примеры: СДО (Moodle, Stepik, …) или чат-бот |
| Плагин |  |
| Плагин для описания предметной области |  |
| Плагин представления закономерностей в Пр.обл. |  |
| Модель студента |  |
| ID студента | Уникальный ID для идентификации студента в Ядре, а именно – в СУ Моделями студентов |
| [2]  Генерация условий (задач) |  |
| [3]  Оценка ответа |  |
| [4]  Выбор (следующего) вопроса |  |
| Стратегия выбора (следующего) вопроса |  |
| [5]  Генерация объяснений | Выполняется по факту допущенной ошибки, используя информацию о конкретной решаемой задаче и конкретном виде ошибки |
| Статистика | По ходу обучения |
| Статистика для преподавателя |  |
| Генерация частично правильного ответа |  |
| Рассуждатель, ризонер, reasoner |  |
| ЧПЯ | Человеко-понятный язык – язык предствления информации в понятном человеку виде: текст на ЕЯ, схемы, диаграммы, … |
| Интеракция | Активность «действие студента - комментарий от системы» в рамках выполнения одного вопроса |

# Задачи на ближайшее время:

1. Создать «Проект» на политеховском/кафедральном **MediaWiki** и подключить участников
2. Создать репозиторий + Issue Tracker (GitHub?)
   1. + Taiga
   2. Какое название проекта взять? **OntoQuiz** подойдёт?
   3. Внести туда задачи (как issues)
3. Подумать над созданием языка для описания закономерностей (в универсальной форме, не зависимой от Пр.обл.)
4. Представление алгоритмов и трасс (в стиле псевдокода):
   1. ! Спецификация самог***о*** используемого языко-независимого псевдокода.
      1. Посмотреть на советы Литовкина Д.В. по форматированию псевдокода на Программировании.
   2. Подготовить карточки (2-3) для опроса студентов с вопросом о предпочтительности формы именования действий алгоритма. Пока варианты такие:
      1. Сквозная(?) нумерация: «<видовое имя>№**N**»;
      2. Случайные существительные (животные, растения, …) без номеров – для элементарных и сложных действий;
      3. Надёргать выражения из реальных open-source проектов (посмотреть codeexamples.com*?*);
      4. Использовать комментарии к желаемым действиям (из тех же открытых проектов?) вместо кода действий (тогда возможны бессмысленные попытки интерпретации).
      5. Для составных действий можно делать составные смысловые имена, например:
         1. Следование –> Тело-функции-factorial
         2. Цикл-по-counter
         3. Развилка-по-is\_admin
         4. Выбор-по-my\_variant
         5. (где **is\_admin**, **counter** и **my\_variant** – переменные, **factorial** - функция)
5. Выбор языков и средств:
   1. для реализации веб-сервиса (критерии: быстродействие, расширяемость, развёртываемость, защищённость, удобство программирования):
      1. С++ (Qt)
      2. + Java
      3. js (Node.js)
      4. ?
      5. Поиск мануала +/ уроков для быстрого старта
   2. Backend рассуждателя (критерии: поддержка средств вывода для наших нужд (в т.ч. возможность перезаписи фактов), выразительность логики, развёртываемость, быстродействие):
      1. OWL +SWRL:
         1. Удалённый сервер Stardog 7 (на Unix) – заявляет быстроту и поддержку современного SWRL (ризонер Pellet 3)
         2. Локальный Pellet 2 (меньшая, но достаточная выразительность) + CLI / интерфейс Java (OwlAPI)
         3. Неизвестные пока другие ризонеры с подержкой SWRL (Например, FaCT++ в виде DLL/.lib)
      2. Prolog:
         1. Открытый SWI-Prolog – (свойства не исследованы)
         2. Другие реализации - ?
6. Введение участников проекта в курс дела:
   1. SWRL
   2. Всем – посмотреть про доверительный интервал (что такое, как понимать) – в 2 вариантах – классический и байесовский.
7. Подумать над системой расчёта статистики по студенту
   1. Форма представления вероятности понимания
   2. Доверительные интервалы для оценки
8. Задача – интерфейс 1 (описание предм.области)
   1. Подсветка синтаксиса в NPP
      1. Служебные слова (словосочетания в кавычках)
      2. Концепты (фор/иф/)
      3. Глаголы
   2. Правила для операторов – в язык - Изменения в правила (Носкин)
      1. Для циклов – явный акт вычисления условия, акт завершения итерации
      2. Акт – это сокращение начала и конца промежутка; простой оператор – сокращение акта с пустотой/элем. действием
      3. Функции делим на пользовательских и библ.(?) – свёрнутых и развёрнутых (при трассировке)
      4. Альтернатива (с **любым** кол-вом веток)- if – if-else – if (развилка / множество альтернатив)
      5. Выбор – switch
      6. Тип цикла ДЛЯ КАЖДОГО (for in/ foreach)
         1. Для ассоциативных контейнеров (мэп, хеш) – посмотреть современные стандарты с++ / java …
   3. jetBrains MPS - ?
9. Вопрос может включать несколько интеракций