

TOWSON UNIVERSITY
OFFICE OF GRADUATE STUDIES

AUTOMATED DIALOGUE SYSTEMS FOR REQUIREMENTS
ELICITATION PRACTICE

By

Erika Marie Boquist

A Thesis

Presented to the Faculty of

Towson University

in partial fulfillment for the degree of

Master of Science

Department of Computer and Information Sciences

Towson University
Towson, Maryland 21252

December, 2013

**TOWSON UNIVERSITY
OFFICE OF GRADUATE STUDIES**

THESIS APPROVAL PAGE

This is to certify that the thesis prepared by _____ [INSERT Student's Name]

ERIKA BOQUIST

entitled_ [INSERT Title of Thesis] _____

REQUIREMENTS ENGINEERING WITH AUTOMATED
DIALOGUE SYSTEMS

has been approved by the thesis committee as satisfactorily completing the thesis
requirements for the degree _ [INSERT Type of Degree] _____

(for example, Master of Science)

Surranjan Chakrabarty
Chair, Thesis Committee

12.13.2013
Date

[Signature]
Committee Member

12-13-2013
Date

[Signature]
Committee Member

12/13/2013
Date

Committee Member

Date

Committee Member

Date

Janet V. DeHoney
Dean of Graduate Studies

2-4-2014
Date

Acknowledgements

Dr. Suranjan Chakraborty served as the thesis advisor and provided a vast amount of expertise regarding software requirements elicitation. His primary research related to requirements engineering processes and understanding socio-cognitive processes underlying the formulation of requirements. For this project, Dr. Chakraborty provided the primary project outline as well as weekly meetings to discuss progress and future improvements of the project.

Dr. Josh Dehlinger is a member for Towson University faculty and will serve as a member of the Thesis Committee as his area of expertise is in the field of Software Engineering. Dr. Dehlinger also aided the project in providing a website front end from his previous project to aid in the development of the chatbot front end.

Atiya Afsana is a doctoral student in pursuit of her Doctorate of Science in Applied Information Technology. She has aided in this project by researching and developing questions for the final chat bot along with their perspective answers. Her doctoral studies may include this project to further her studies.

Dr. Siddharth Kaza is also a member of Towson University faculty and will serve as a member of the Thesis Committee with experience in Knowledge Discovery. Dr. Kaza also assisted in weekly meetings to provide advice for GUI implementation and possible future server-side implementation.

ABSTRACT

AUTOMATED DIALOGUE SYSTEMS FOR REQUIREMENTS ELICITATION PRACTICE

Erika Marie Boquist

Background:

Requirements gathering is a vital part of the software development lifecycle. The act of requirements elicitation revolves around prompting prospective clients with questions that will result in an efficient set of requirements. The questions an analyst, software engineer, developer, or project manager asks and the way that they follow up on those questions to probe clients for requirements provide information to aid in the requirements elicitation process and a system's development as a whole.

Problem Definition:

The act of requirements elicitation is difficult and requires an immense amount of skill and attention to detail to not only draw all requirements from the clients, but to also provide a solid basis for system development. Choosing a questioning methodology is vital to comprehensive coverage of all categories of requirements.

Approaches:

The intention of this project was to develop an automated dialogue system that would allow users to interact with and discover appropriate questions for requirements elicitation. The notion of a question answer (QA) system would give users the ability to engage in a natural language, human-like conversation, to simulate an environment in which an analyst could ask questions to elicit system requirements.

For this project, two types of dialogue systems were implemented. The first interaction was an AIML closed-domain chatbot allowing for open-ended set of inputs, whereas the second closed-domain iteration implemented a closed set of acceptable inputs.

Each iteration accounted for all requirement categories and possibility of questions based on a prompting technique that was researched and deemed to be a best fit for the system descriptions that were included in the dialogue system's domain.

Results:

In conducting a pilot study on the first iteration chatbot, a number of limitations regarding AIML's handling of an extremely open-ended variety of questions specific to a requirements elicitation domain were discovered.

The nature of the second iteration's restricted set of inputs guaranteed 100% accuracy in the system's responses, but removed the user's ability to devise their own questions.

Conclusions:

This project was an introduction to developing a much larger system aiding in the training of students pursuing and understanding of requirements engineering and the requirements elicitation phase of the software development lifecycle. The dialogue system implementations, results and findings of this thesis will be used in a doctoral study expected to continue.

Table of Contents

TABLE OF CONTENTS.....	VI
TABLE OF FIGURES	VIII
TABLE OF TABLES.....	IX
INTRODUCTION.....	1
PROJECT OVERVIEW	1
OBJECTIVES AND GOALS.....	1
BACKGROUND RESEARCH	3
REQUIREMENTS ENGINEERING PROMPTING EFFECTIVENESS BACKGROUND.....	3
TECHNOLOGICAL BACKGROUND	4
EXPLORED TECHNOLOGIES	5
<i>AIML and ALICE.....</i>	<i>5</i>
<i>JavaScript, jQuery, and Hashmaps</i>	<i>6</i>
INTERACTIVE DIALOGUE SYSTEMS.....	8
ITERATION 1	8
<i>Knowledge Base Implementation</i>	<i>8</i>
<i>First Generation Bot.....</i>	<i>9</i>
Front End Development	9
Back End Development.....	11
Testing Results and Analysis	12
<i>Second Generation.....</i>	<i>13</i>
Front End Development	13
Back End Development.....	15
Testing Results and Analysis	15

Usability Study Results and Analysis	16
Linguistic Issues	19
Synonym Issues	21
Survey Results	22
ITERATION 2	23
<i>Front End Development</i>	23
<i>Back End Development</i>	25
<i>Proposed Testing Results and Analysis</i>	25
Development	25
Expected Results	26
CONCLUSION	27
CHALLENGES AND LIMITATIONS	27
FUTURE DIRECTION AND DEVELOPMENTS	28
<i>Developments</i>	28
<i>Implementations</i>	28
Front End Developments	28
Back End Developments	29
APPENDICES	30
APPENDIX A – QUESTION SET FOR PROTOTYPE	30
APPENDIX B – PROTOTYPE STUDY RESULTS	32
REFERENCES	33
CURRICULUM VITA	34

Table of Figures

Figure 1 - Semantic Prompting Technique	4
Figure 2 - jQuery Autocomplete	7
Figure 3-Trial Bot Splashscreen.....	10
Figure 4-Trial Chat Bot.....	10
Figure 5-Trial Chat Bot Topic	11
Figure 6- Trial Chat Bot Topic Templates	12
Figure 7 - Second Generation Chatbot	14
Figure 8 - Pilot Study Instructions.....	17
Figure 9 - Pilot Study Questionnaire.....	17
Figure 10 - User Input Handling	19
Figure 11 - Question Diagram A.....	20
Figure 12 - Question Diagram B	20

Table of Tables

Table 1 – AIML Category	5
Table 2 - jsHash Implementation.....	7
Table 3 - Requirement Categories and Questions	9
Table 4 - Grades Patterns.....	12
Table 5- Q&A Retrieval	14
Table 6 - Synonym Lookup Code	21
Table 7 - Questionnaire Results	22
Table 8 - Mongoose Schema	29
Table 9 - Prototype Study Results	32

Introduction

Project Overview

Analyst questioning is a critical part of requirements elicitation. The questions an analyst, software engineer, developer, or project manager asks and the way that they follow up on those questions to probe clients for requirements provide information to aid in the requirements elicitation process and a system's development as a whole. Improving developer questioning can be a catalyst for the software development process.

The act of requirements elicitation is difficult and requires an immense amount of skill and attention to detail, to not only draw all requirements from the clients, but to also provide a solid basis for system development. As one of the most critical aspects of information systems development, numerous studies have been conducted discussing prompting techniques to acquire a requirement fit for a specific category (Browne & Rogich, 2001). Choosing a questioning methodology is vital to comprehensive coverage of all categories of requirements.

In an attempt to assist and further the requirements elicitation phase, we proposed the following:

- Designing an interactive dialogue system to model prospective client responses and respond in real-time to prospective developers' questions regarding a system's requirements.
- Developing a knowledge base for the interactive dialogue system by researching types of questions and their corresponding requirement categories to program potential responses.
- Piloting the system through integration with existing assignments as a part of a graduate Requirements Engineering course.

Objectives and Goals

The intention of this project was to develop an automated dialogue system that would allow users to interact with to discover appropriate questions for requirements elicitation. The notion of a question answer (QA) system would give users the ability to engage in a natural language, human-like conversation, to simulate an environment in which an analyst could ask questions to elicit system requirements.

There were three goals associated with this thesis:

- Gaining an understanding of the background research for implementing a dialogue system
- Developing a prototype
- Running a pilot study on the prototype

The research objectives included a substantial study of background research regarding best practices for requirements elicitation. Additionally, we acquired knowledge to fully implement a dialog system prototype that required extensive study of QA systems. The objectives of this research phase were the following:

- Construction of a prototype dialogue system knowledge base for the proposed knowledge domain relating to requirements discovery
- Acquire an understanding of the Zipf's law and curve and their applicability to the requirements domain.

Background Research

Requirements Engineering Prompting Effectiveness Background

A substantial portion of the engineering research was on systems analyst questioning techniques. This included understanding the content to be used in developing the knowledge base of the dialogue system. For both the trial and final systems, all information was gathered to successfully implement the dialogue system.

The first step was gathering all information or requirements that the user would be polling. The requirements necessary for complete system coverage were the following:

- Goal level requirements
- Process level requirements
- Task level requirements
- Information requirements

Once a collection of requirements was established, the Semantics prompting technique was employed to develop questions according to events, states, conditions, actions, agents, and goals (Browne & Rogich, 2001). The figure below was adapted from a Brown and Rogich figure and displays types of prompts to acquire requirements in each category that would serve as guidelines for handling user inputs.

Goals What are the system goals? How is each goal attained? Why is each goal important? What indicates that each goal is achieved?	Events What events affect the system? What are consequences of each event occurring? What causes each event to occur? What goal does each event fulfill?
Agents Can you name a person or department involved with the system? What role does each play? What are his or her goals? What agent has opposing goals?	States or Conditions What states or conditions affect the system? What causes or enables each state? What are the consequences of each state being present? What goal does each state support?
Actions Can you name the actions involved in the system? How does a person perform each action? What prevents a person from being able to perform each action? What goal(s) does each action satisfy?	

Figure 1 - Semantic Prompting Technique

These questions formed the basis of the question set developed for the dialogue system to ensure acquisition of requirements for each of the categories. The questions set developed will be discussed further in this document.

Technological Background

The proposed system was implemented as a conversational agent using technological affordances provided by question answer (QA) systems. This type of system is a type of natural language dialog system (NLDS) that allows users to interact with a system with natural language queries and responses. In a 2003 study, authors Moldovan, Pasca, Harabagiu and Surdeanu discussed the performance of a QA system as a combination of question complexity and the difficulty of answer extraction. In that study, it was determined there were 5 classes of QA systems: processing factual information, simple reasoning, and inferring answers from multiple document combination, interactive systems, and systems capable of analogical reasoning (Moldovan et al. 2003).

Explored Technologies

AIML and ALICE

The initial system at the core of this proposal was based off of the ALICE chat bot, developed in 1995 by Richard Wallace and the Alicebot free software community. ALICE (Artificial Linguistic Internet Computer Entity) is a class 1 QA system that employed AIML, or Artificial Intelligence Mark-up Language (Wallace, 2003). The chat bot was created as a conversational software agent used to interact with users in natural language. While conversations in natural dialog would seem endless and impossible to simulate, Zipf, a contemporary of Turing, discovered a characteristic of human language that would greatly affect the advancements of artificial intelligence (Wallace, 2003). The idea of Zipf's law was that sentences are distributed throughout a conversation according to a specific curve. Zipf discovered that of all the words, only about two thousand are used first in a sentence and that the frequency of a word was inversely proportional to its rank in a frequency table. The frequency of each word is related to the ranking of each of the subsequent words. He found that while words that could be articulated are infinite, only 1800 words covered 95% of first words that are ever spoken (Goh, Fung & Depickere, 2007).

Shawar (2011) described two types of chat bots: open-domain and closed-domain. The distinction between an open or closed domain system lies in its content. In an open domain system, the content is everything related to general knowledge. In contrast, a closed domain system has a very specific domain. The chat bot developed for this proposal was a closed-domain bot with domain-specific knowledge relating to software requirements discovery.

ALICE took advantage of the both human and machine readable behavior of XML (Extensible Markup Language) by defining rules for properly encoding its files. The combination of various documents conforming to a set of rules would then be handled by an interpreter. AIML is a dialect of XML with its interpreters available under the GNU General Public License and hosted on Google Code (Wallace, 2013).

The focus of this project was the knowledge base of an AIML chatbot. The knowledge base of this chatbot was a collection of units framed by tags specific to AIML. Each of these units would address content specific to our system's domain. Paired with the AIML interpreter and an HTML front end, the implementation of our chatbot would be complete (Wallace, 2003).

The fundamental unit of knowledge, the category, is devised of input questions, called patterns, which result in a response, or template. These AIML categories are denoted as:

```
<category>
<pattern> HELLO </pattern>
<template> Hi there!</template>
</category>
```

Table 1 – AIML Category

The chat bot receives a pattern, which is parsed into the best-fit category, and provided with a response, or template; each of these domain-specific units are stored in a knowledge base.

Alicebot was an attempt to bridge the gap between the slow information exchange rate of human dialogue and the significantly faster computer communication. Proof of ALICE's progress is the increasing lengths at which conversation can be maintained with the bot (Wallace, 2003). Each year a contest called Chatbot Battles is held challenging participants to develop chatbots that stimulated longer conversations and conversations akin to natural language of a 7th grader. Last year's contest winner's chatbot maintained a 15 minute conversation with the judges (Worswick, 2012).

The implementation research consisted of understanding how to implement a closed-domain chat bot with the use of HTML and AIML. HTML required very little background research, as this was already a familiar language. Similarly, AIML is based on XML so extensive research was not necessary and only required knowledge of nuances specific to AIML.

The chatbot used the Pandorabots open-source web service which enabled the development and hosting of the chatbot free of charge. Pandorabots is the largest chatbot community on the Internet and is constantly evolving ("Pandorabots," 2002). The chatbot hosted on Pandorabots had an initially empty knowledge base that we built upon. Additionally, we developed a website front-end for increased human-computer interaction satisfaction.

The primary source of AIML documentation was Richard Wallace's *The Elements of AIML Style*, which provided history and implementation guidance for the chatbot prototype.

JavaScript, jQuery, and Hashmaps

The second candidate technology that was explored was a combination of JavaScript, jQuery and hash maps to provide users with an input dialog enhanced with autocomplete to ensure appropriate questions as well as their corresponding answers.

The use of HTML and JavaScript became self-evident, as JavaScript is an object-oriented scripting language used on the Internet. JavaScript has an enormous number of libraries, including jQuery. One of the many capabilities of jQuery is the auto-completion. A listing or pre-populated values displaying upon matched user input would limit the types of input allowed. JQuery's Application Program Interface (API) provides for seamless integration with the JavaScript backend (Swedberg, 2013).

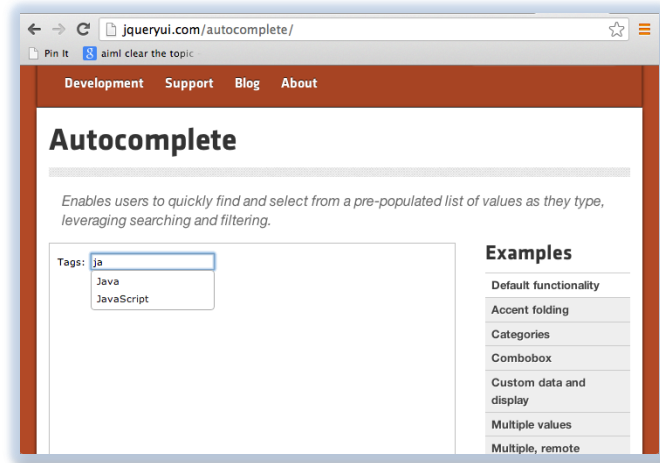


Figure 2 - jQuery Autocomplete

With the need for storage of inputs and their corresponding outputs, a hash table created at runtime sufficed. Hash tables or hash maps are data structures designed to implement associative arrays. The idea is to map keys to values, or in this system, questions to answers. A simple JavaScript hash table, with the ability to add, remove, modify, and look up values by their keys, provided all the requirements needed for this type of dialogue system. The example below demonstrates a simple jsHash implementation.

```
/**
 * @desc demonstrates the jsHash implementation
 * @author Erika Boquist
 *
 */
var questions = new jsHash(),
    questionArray = [
        {
            question: "What is the goal?",
            answer: "The goal is to have 120% profit."
        },
        {
            question: "Why is the goal important?",
            answer: "To increase company stability."
        }
    ];

//push the questions from the question array to the available tags
//and the entire set of QAs to the hash
for(var i = 0; i < questionArray.length; i += 1){
    availableTags.push(questionArray[i].question);
    questions.setItem(questionArray[i].question,
questionArray[i]);
}
```

Table 2 - jsHash Implementation

Interactive Dialogue Systems

Iteration 1

The trial dialogue system was a chatbot with a well-defined AIML architecture hosted on Pandorabots and an appropriate GUI for users to interact with. This trial bot had a closed domain knowledge base which was populated with data from the Fall 2013 Software Requirements Engineering course syllabus; topics were limited to the following: course, exams, grades, instructor, papers, test, textbook.

Two generations of trial bots were necessary to meet all the development needs while only one knowledge base had to be developed.

Knowledge Base Implementation

As the trial chat bots domain was specific to the course syllabus, the devised question set that needed be accounted for in the knowledge base of the chat bot for each requirement category are displayed in the tables below:

Goal Level Requirements	Process Level Requirements
What is required to pass the course?	What do students have to do to pass this course?
What is considered passing?	What are the prerequisites for this course?
How is the course graded?	What assignments are in this class?
What are the time constraints for this course?	What projects are in this class?
What kind of support system is there for this course?	What exams are in this class?
Who is involved in this course?	What papers need to be written in this class?
What is this class about?	What is the weight of the (assignment, project, exam, and paper) to the overall class?
What is the expected outcome from this course?	When does the class meet?
What outside support is there for this course?	Where does the class meet?
	What textbooks are required for the course?
	What other suggested readings are there?

Task Level Requirements	Information Level Requirements
When is (assignment, project, exam, and	Where will the grades be stored?

paper) due?	
How is the (assignment, project, exam, and paper) to be executed?	Where will course information be hosted?
How is the (assignment, project, exam, and paper) graded?	How is course information to be accessed?
What purpose does the (assignment, project, exam, and paper) have?	How are the students to communicate with the professor?
Who is required to perform the (assignment, project, exam, and paper)?	How are students to submit assignments?
What outside help is there for the (assignment, project, exam, and paper)?	How will students receive their graded (assignment, project, exam, and paper)?
What are class (room) policies?	

Table 3 - Requirement Categories and Questions

These questions were expanded into a set of individual and common language questions which are listed in Appendix A.

This set of questions would elicit all requirements related to the course syllabus, having employed the semantic prompting technique, ensuring each question began with a “who, what, why, where, when, and how” question word. Additionally, the set of questions would satisfy acquisition of requirements to fit each of the requirement categories.

First Generation Bot

The first generation bot consisted of a simple webpage front end and AIML backend.

Front End Development

The front end provided a brief explanation on proper questioning (See Figure 4). This page included information from the Browne and Rogich article discussing prompting techniques for acquiring goal, process, task, and information level requirements. It also discussed the importance of prompts from the interrogatory technique (Browne & Rogich, 2001).

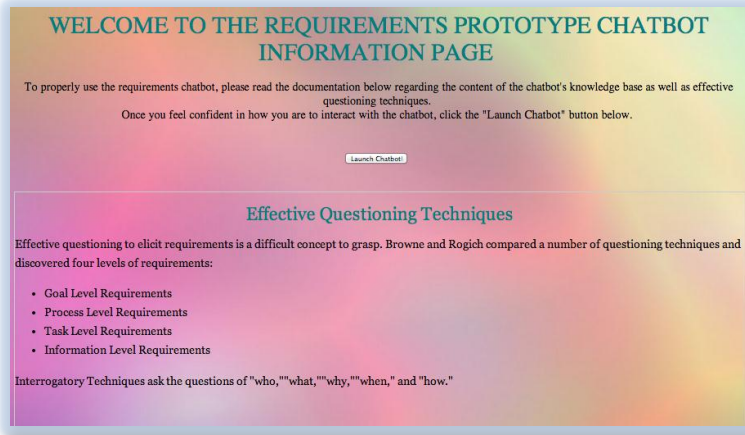


Figure 3-Trial Bot Splashscreen

Once the user was sufficiently educated in proper prompting techniques, a “Launch Chatbot” button’s activation would launch a nested webpage hosted on the Pandorabots website. This page provided sample appropriate questions such as “Who is the instructor” and “Where does the class meet.”

The user could then input their prompt in the input text area defined by “You:” and the chat bot would respond in the text area below the “BOT:” text (See Figure 5).

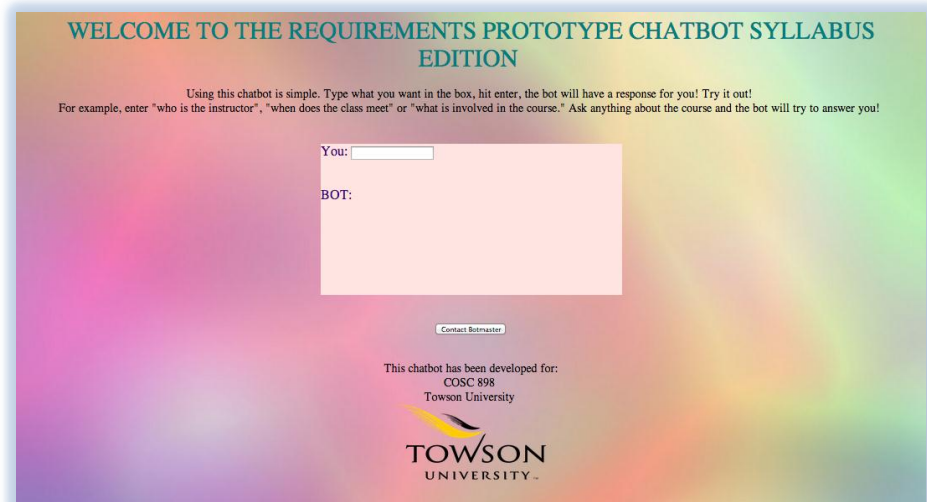


Figure 4-Trial Chat Bot

Additionally, each of the web pages included a “Contact Botmaster” button to send emails to:

eboqui1@students.towson.edu.

Information regarding the purpose of the chat bot development as well as a link to Towson University’s homepage was provided as a live link on the Towson logo. The

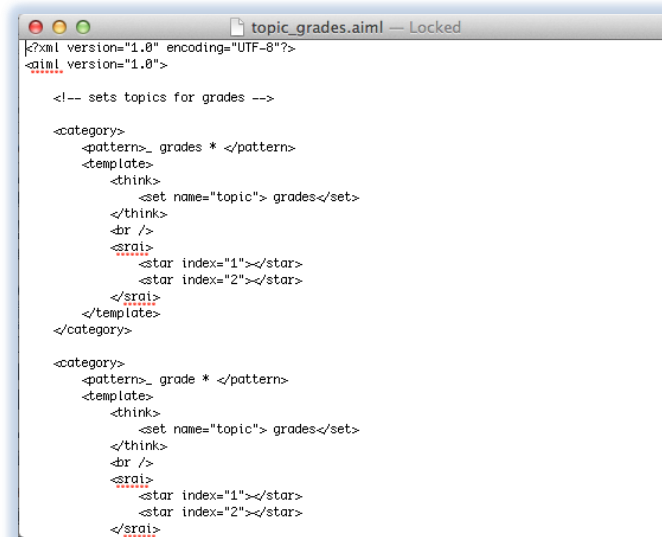
logo was included according to Towson's brand mark standards and download page, which can be found at:

<http://www.towson.edu/creativeservices/logos.asp>

Back End Development

Research of AIML structures provided very little information for proper architectures. The majority of sample code displayed all AIML patterns in a single file. To maintain a well-organized backend, a Towson University previous chat bot implementation architecture was mimicked to filter on specific topics.

The backend of this trial chat bot was AIML with an architecture that would parse the input, filtering on a specific topic. Topics that were successfully parsed are the following: assignments, assistants, course, exams, grades, instructor, papers, projects, test, and textbook. Figure 3 displays the introduction to filtering the input based on the topic "grades" being present.



```

<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
  <!-- sets topics for grades -->
  <category>
    <pattern>_ grades * </pattern>
    <template>
      <think>
        <set name="topic"> grades</set>
      </think>
      <br />
      <srai>
        <star index="1"></star>
        <star index="2"></star>
      </srai>
    </template>
  </category>

  <category>
    <pattern>_ grade * </pattern>
    <template>
      <think>
        <set name="topic"> grades</set>
      </think>
      <br />
      <srai>
        <star index="1"></star>
        <star index="2"></star>
      </srai>
    </template>
  </category>
</aiml>

```

Figure 5-Trial Chat Bot Topic

Once the topic was discovered, various templates would be combined into a single file for each topic.

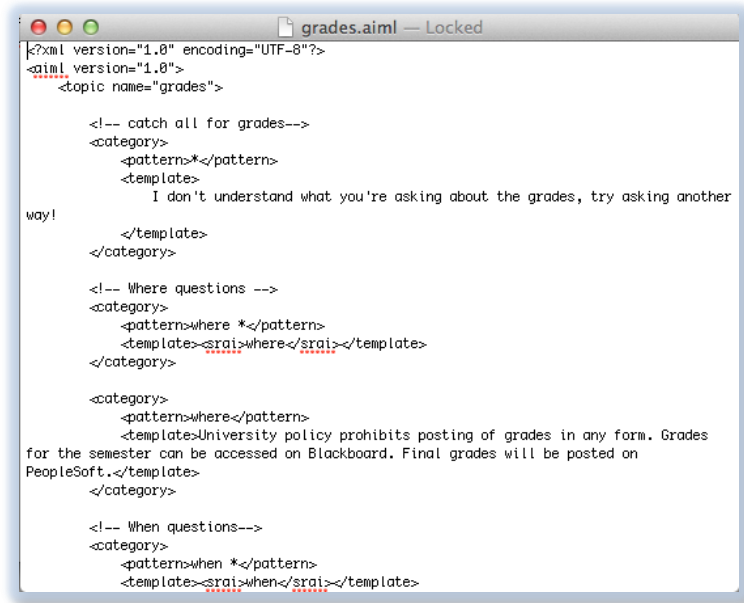


Figure 6- Trial Chat Bot Topic Templates

Figure 4 displays the set of templates that would be available to an input in which the topic “grade” was captured.

Testing Results and Analysis

Testing of the first trial bot was strictly on a developer level. Imitating a prospective user’s interaction, each of the questions from Appendix A were input into the system. The inputs and corresponding outputs were then recorded for analysis. A number of issues were discovered during analysis.

The first issue that was found was AIML’s need for extensive redundancy. To successfully capture a topic there had to be three patterns for every question. For example, to capture the topic “grades,” the following three patterns were required:

<pre> <pattern>_ grades * </pattern> <pattern>_ grades </pattern> <pattern> grades _</pattern> </pre>

Table 4 - Grades Patterns

The use of the wildcards above account for following three different sentences, respectively: “What grades can I expect,” “What are grades,” and “Grades are what.” This issue with this effort of redundancy is there is a greater likelihood of human error in missing a pattern, size of source code, and most importantly a higher cost of running code.

The second issue in this architecture design was recursive patterns being overwritten by wildcards, specifically the underscore. An example of this was in the question, “What percentage of the grade are projects?” In this case, the topic was “projects” was set, however when “grade” was parsed, the topic was reset according to the alternative “grading” topic. This was causing incorrect patterns to be matched resulting in either wrong output, or no output at all.

The third issue was quite a number of synonyms were required to capture variations on the same question that a user would submit. Not only did we have to include synonyms such as “lecturer,” “teacher,” and “professor” for the word instructor, but we also had to include singular and plural variations on the same term.

The topic parsing files were enormously bloated and a significant amount of time was spent including extra patterns and synonyms, time that could have been better spent on furthering project implementation rather than in a thesaurus.

Second Generation

The second-generation bot evolved the front end only slightly, while the back end changed substantially.

Front End Development

The front-end goals were to removing the splash screen providing details on requirement elicitation and to add to provide conversation logging.

Initially, it was suggested to update the front end to log conversations by adapting an existing front end. This front end included not only conversation logging but the capability for exporting the logs to a file the user could download. The project was modified and deployed to Google App Engine, however, when running the project, the Pandorabots service could not be reached, resulting in an HTTP 503 error.

Due to time restraint, the solution for conversation logging was to incorporate an additional log area on the previously developed chat bot front end (see Figure 8).

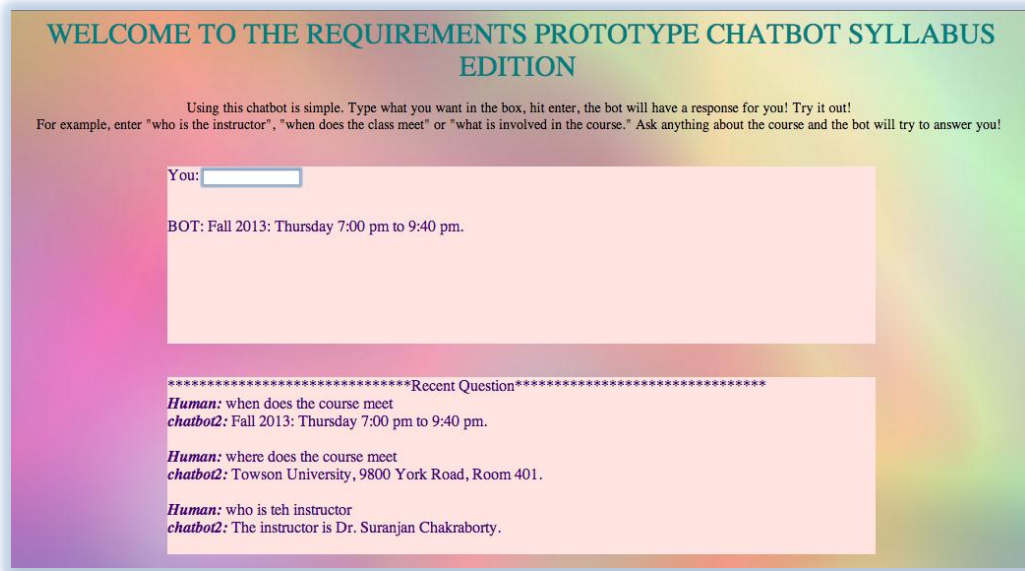


Figure 7 - Second Generation Chatbot

Implementation took advantage of AIML's history by creating a template with condition polling the history's request indices. The code sample below shows a complete template for retrieving the most recent question and answer. The entire implementation included a number of <think> tags as well as <conditions>.

```
<template>
  <think>
    <set name="_history_1"><request index="1"/></set>
  </think>
  <condition name="_history_1" value="*">
    <i><b>Human:</b></i> <request index="1"/><br/>
    <i><b><bot name="name"/>:</b></i> <response index="1"/><br/>
    <br/>
  </condition>
</template>
```

Table 5- Q&A Retrieval

While the logging successfully displayed the user's most recent questions and answers, AIML currently supports logging only 20 interactions. This required the users to copy and paste several times their 20 most recent interactions. For a second-generation trial bot, this wasn't optimal but sufficed.

Back End Development

The architecture for the new back end was similar to the first generation trial bot in that the entry point filter.aiml file would parse the input, capturing a topic that would then be passed to separate files pertaining to each topic.

The difference in this generation bot was that topics were not based on tangible concepts such as “instructor,” “textbooks,” or “course.” These topics corresponded with prompts defined in the Browne and Rogich study: “what, where, when, how, and who.” Each of these prompts was associated with a variety of questions:

- What – Prompts for requirements regarding an item or concept.
- Where – Prompts for requirements regarding a location.
- When – Prompts for requirements regarding a time.
- How – Prompts for requirements regarding a process’ execution.
- Who – Prompts for requirements regarding a person.

Each of these prompts corresponded to a separate AIML file, which then included a second level of filter based on subtopics, or types. These types were associated with the various object of each input. For example, when asking when something will be graded, the object could be: papers, projects, exams, or assignments. Type filtering reduced a significant amount of redundancy in similar pattern matching.

Testing Results and Analysis

Testing of the second-generation chatbot was similar to the first chatbot testing. Inputs were from the question list in Appendix A, and the questions and system’s answers were recorded. The second-generation chat bot greatly reduced the number of recursive collisions; however more nuances of AIML became apparent.

Order of input parsing plays an important role on how AIML is executed. An example is how the following sentence is parsed: “What is the purpose of the projects?” There are two categories checking for the keywords “purpose” and “projects.” In this example, the category referencing “purpose” would always execute first. This created a problem as a second input question, “What is the purpose of the exams?” While the aforementioned subtopic, or type, is set to the “projects, papers, instructor, etc.” the “purpose” will always execute first resulting in an empty response from the chat bot. The solution for this parsing issue was modifying the input from “What is the purpose of the exams” to “The projects have what purpose?” This was expected to cause an issue for user interaction, as input wording would be critical, and something that could not be guaranteed among a variety of users.

Of the question created for the chat bot, the following questions had to be adjusted:

1. ~~What percentage of the grade are projects?~~ What is the weight of projects on the final grade?

2. ~~What percentage of the grade are papers?~~ What is the weight of papers on the final grade?
3. ~~What percentage of the grade are exams?~~ What is the weight of exams on the grade?
4. ~~What percentage of the grade are assignments?~~ What is the weight of assignments on the grade?
5. ~~What is the purpose of the assignments?~~ What is the assignments purpose?
6. ~~What is the purpose of the projects?~~ What is the projects purpose?
7. ~~What is the purpose of the exams?~~ What is the exams purpose?
8. ~~What is the purpose of the papers?~~ What is the papers purpose?
9. ~~What outside help is there for the assignments?~~ What assignments help is there?
10. ~~What outside help is there for the projects?~~ What projects help is there?
11. ~~What outside help is there for the papers?~~ What papers help is there?
12. ~~What outside help is there for the exams?~~ What exams help is there?

Once these inputs were modified, the entire question set correctly responded to by the chat bot.

Usability Study Results and Analysis

A short usability study was performed on the second-generation trial chat bot. The study was conducted in the Towson University Software Requirements Engineering graduate course. The study included 14 participants of which the age, race, and gender were not documented and therefore unknown.

Participants were provided a short instruction sheet with directions for interacting with the chat bot (see Figure 9).

INSTRUCTIONS

Introduction

This exercise is designed to sensitize you with the challenges inherent in eliciting an understanding about a problem domain through questioning. As a part of this exercise you will interact with a chat bot that has been programmed with a knowledge base about the syllabus for this course. You are expected to ask the Chatbot a series of questions and develop an understanding of the syllabus from its responses.

Link

To interact with the interface, please click on the following link
<http://www.pandorabots.com/pandora/talk?botid=901dc0083e3474d8>

Recommendations for Interactions:

To interact with the Chatbot system, you need to begin your questions with "who", "what", "when", or "how". The Chatbot also expects certain specific structure for the questions, so you may need to try out variations of the questions in order to get the answers. **For example:** if you want to query the Chatbot about the instructor, it would be more appropriate to ask "Who is the instructor?" rather than "What is the instructor's name?" We would be using the information gained from your interaction to improve the Chatbot, so please copy the interaction transcript that is displayed in the lower half of the screen into a word document and upload it using the Assignment Link when you are done. The interface only records the *10 most recent set* of interactions, so please copy these into the word document as you keep working.

Figure 8 - Pilot Study Instructions

Once the participants had completed and submitted their chat logs to the assignment link on blackboard, they were then given a questionnaire to complete (See Figure 10).

COSC 601/AIT641- Software Requirements Engineering

Questionnaire

Please respond to the following questions to your best ability.

1. What did you like about interacting with this Chatbot?
2. What did you not like about interacting with the chat bot?
3. What would you like to share about using this Chatbot?

Figure 9 - Pilot Study Questionnaire

Each of the inputs were parsed and placed in a table with twelve categories. If the system response was correct, the correct category would be marked. If the response was incorrect, the appropriate categories would be checked. These categories were as follows:

- Prompt – For every input, the prompt used was marked in this category. This would be “Who, what, where, etc.” or could be empty if a prompt was missing.
- Topic- The topic that was matched or should have been matched in the input was marked in this category.
- Sub Topic- Sub topics that were matched or should have been matched in the input was marked in this category.
- Missing Synonym – An appropriate synonym that was used in the input but was neglected in the chatbot’s knowledge base was marked in this category.
- Missing Descriptor – A descriptor was categorized as the subject of the question and a modifier of the topic in the response. Descriptors that should have been included in the chatbot’s knowledge base were placed in this category.
- Poor Ordering – In the event that sub topic was matched prior to the topic causing an incorrect response by the system, this category was selected.
- Previous Topic Matched – Topic’s that were set when parsing a pattern, would persist until a second topic was set. If a topic a subsequent topic was not properly set resulting in responses related to the previous topic, this category was marked.
- Missing Prompt – This category was marked if a user neglected to include any prompt.
- Incorrect Response – This category was marked if the behavior of the chatbot could not be determined or had no response.
- Irrelevant Question – This category was for questions outside the scope of the knowledge base or were not complete sentences.
- Composite Prompt – A composite prompt was marked as the category if the user’s input contained more than one prompt.
- Correct – Correct interactions were marked in this category.

For a complete table listing of all conversation logs during this study, see the table in Appendix B.

The results of the study displayed a number of general errors:

- Many of the students did not save the transcript logs correctly resulting in the input and chat bot outputs not matching.
- There were an enormous number of synonyms missing from the knowledge base, resulting in a number of patterns that could have matched, failing.
- A large number of inputs were outside the scope of the chat bot’s knowledge base, such as plagiarism, exam preparation, reasons for teaching software engineering, and many more.
- Many students did not use proper English to format their questions.
- Many students entered “garbage” inputs completely unrelated to the syllabus.

The table below displays all the user inputs and a breakdown of their responses. Of the total number of inputs recorded, only 21.5% were answered correctly by the chatbot. 26% of the inputs included synonyms that were missing from the knowledge base, 10% of the questions neglected to include a question word, 26% of the questions irrelevant to the scope of the project, 20% of the questions included descriptors not considered in the knowledge base, with the remaining questions falling into states where the chatbot could not respond at all.

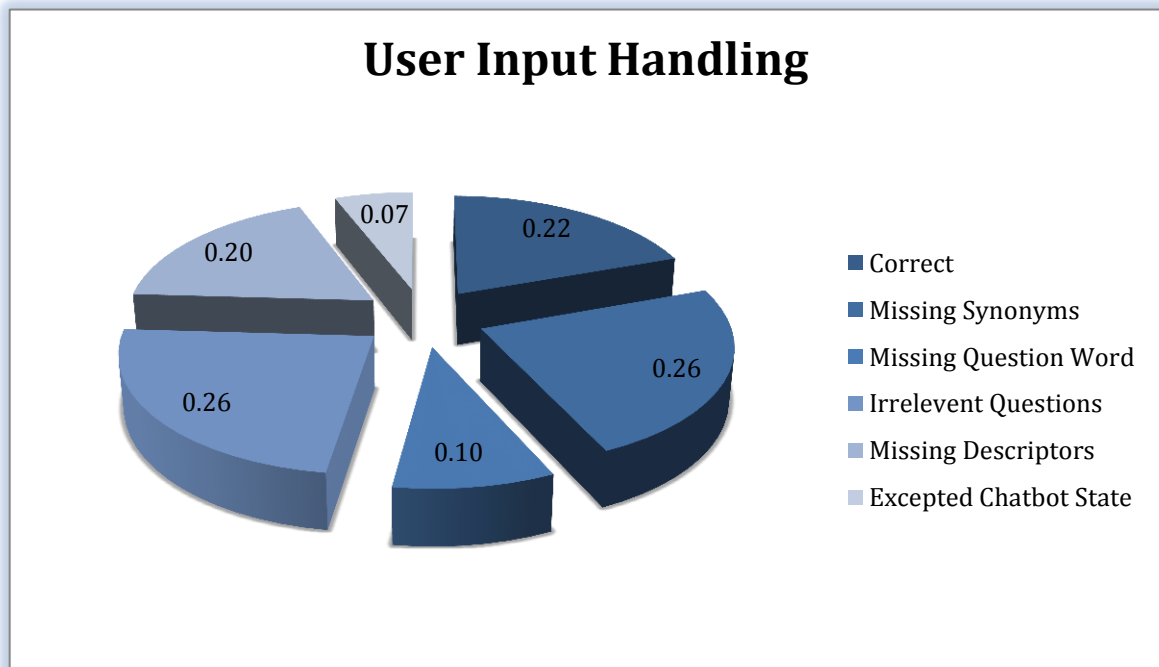


Figure 10 - User Input Handling

Linguistic Issues

The most common variation on a planned question was on the base question “Who is the professor?” Many students asked the same variation of “What is the professor’s name?” These two questions appear the same as they would respond with the same answer, however, the chat bot failed to answer the question regarding the professor’s name. Diagramming these two sentences provided insight as to why these are two different sentences completely (“Sentence Diagrammer,” 2009).

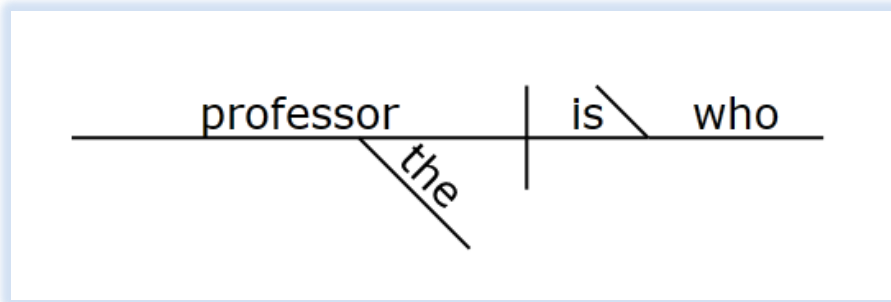


Figure 11 - Question Diagram A

The first and expected input defined each part of speech as the following:

- Who – Pronoun – question word, subject complement noun
- Is – Verb – Verb of question clause
- The – Adjective – Adjective modifier
- Professor - Noun - Subject of question clause

The second sentence diagrams very differently:

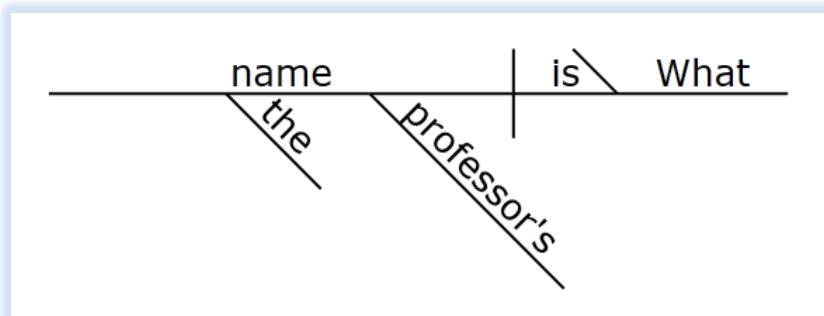


Figure 12 - Question Diagram B

Each part of speech, with the exception of “The” and “is”, behave very differently:

- What - Pronoun – question word, subject complement noun
- Is – Verb – Verb of question clause
- The – Adjective – Adjective modifier
- Professor’s - Noun – Adjective modifier(possessive)
- Name - Noun - Subject of question clause

In the first case, the object of the sentence is “professor,” paired with “Who,” would properly respond with who the professor was: a person. In the second case, the object is “name,” which paired with the “What” would respond with a thing’s name. Without the identifying adjective of “professor” there would be no way of determining who, or what’s, name the user is attempting to retrieve.

This caused a serious issue for the chat bot as the template relating to a person would be in the who.aiml, and a name would be in the what.aiml. A topic of “professor” would have to be set, followed by the sup topic, or type, for the name. Furthermore, the structure did not and was not prepared to house possessive forms of each topic.

This issue led to an investigation of linguistics and sentence parsing. If we could successfully parse a sentence and extract all the sentence parts, we could then set topics and layers of subtopics according to each particle of speech; an algorithm that could normalize the sentence into usable objects containing properties to associate with each topic. Indeed, there were sentence algorithms used for semantic normalization (Hart & Goertzel, 2008).

The science of linguistics is vast, even in a basic overview and far exceeded the scope of this project (Gasser, 2009). Additionally, a majority of the students interacting with the system were not native English-speakers, resulting in sentences that were incomplete, structurally incorrect, misspelled, and convoluted. As time was running out, it was decided to reduce the input complications by limiting the number of inputs to a finite list of questions that the user could choose from.

Synonym Issues

Addressing the second issue of lack of synonyms, the prospective solution was to incorporate a text file dataset to quickly and easily check words inputted from the user and not only verify they were words, but to also return the highlighted word for that category. For instance if the user had entered the word “instructor” but the chat bot used “professor,” a simple check could verify that “instructor” was a synonym for “professor” and would then return “professor.”

```
var result,
word = {word: "instructor", type: "noun"};
if(word.type === "noun"){
  result = checkSynonym(word.word);//calls some function that references a dataset to return the word
  that should be used. "instructor" return "professor"
  if(result){
    switch(result){
      case "professor":
        //do something
        break;
      case "textbook":
        //do something
        break;
      case "course":
        //do something
        break;
    }//end switch
  }else{
    return "This word is not acceptable to the program...not found in dataset";
  }
} //end if noun
```

Table 6 - Synonym Lookup Code

The above code sample displays how instructor would be handled and returned as professor for ease of processing and reduction of AIML. The three variations of handling topics with wildcards would only be needed for one word greatly reducing the size of each AIML file.

Survey Results

The survey results are displayed in the table below. The results were as expected, displaying a mixture of positive and negative responses depending on what set of questions were used.

Question #	Question	Responses
1	Chat Bot Pros	<ul style="list-style-type: none"> • fast • easy to use • easily answered • satisfactory answers • easy to follow • very specific to the questions • started out interesting • sounds intelligent • sometimes provides more info that expected
2	Chat Bot Cons	<ul style="list-style-type: none"> • answers are irrelevant • not smart enough • not enough synonyms • changing one word would give different answers • questions limited to who, when, what, etc. • hard to get worded for questions • sometimes don't get any answer • don't provide many answers • keyword recognition • very frustrating • does not read the entire questions • misinterpret the questions • did not understand simple ones • structure is rigid
3	Comments	<ul style="list-style-type: none"> • keep adding keywords • rephrase the questions • keyword recognition • submit button • doesn't care about predicate "it" 'the', etc. • helpful to have some sample questions,

Table 7 - Questionnaire Results

Future studies would benefit from correlating which user log correlated with survey results to determine credibility of user input as a significant amount of questions were invalid.

Iteration 2

Due to the increasing linguistic difficulties regarding text parsing for the initially proposed automated dialog system, it was suggested that the alternatively researched architecture for the system should be implemented. This architecture affected the front and back ends of the dialog system.

The suggestion was that a closed set of prompts should be established within the question-answer (QA) system to prevent users for inputting arbitrary information as well as to avoid the linguistic difficulties discovered in the previous chatbot system.

Front End Development

Initially, the suggested alternative front-end procedure was to provide a graphical user interface (GUI) that would contain drop down menus for the users to select and input a predefined set of prompts (See Figure 16).

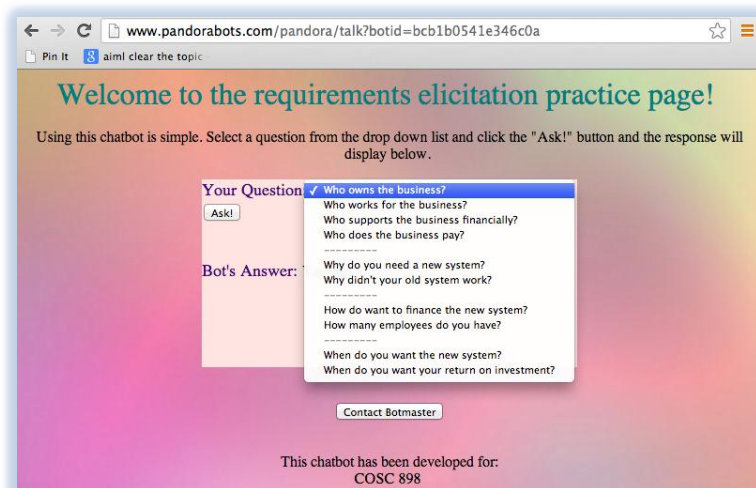


Figure 13 - Iteration 2 Initial Front End

However, upon further review, it was decided that such an explicit set of prompts would nullify the reason for the QA system in its entirety. As such, it was suggested that an auto-complete input form would be appropriate as to wait for the user's initiation prior to displaying prompting suggestions.

For the final GUI, it was decided to take a very simple approach with very little background distraction and a simple interface to provide the user with information on how to use the program as well as a way in which the user can save the results of their QA session (See Figure 17).

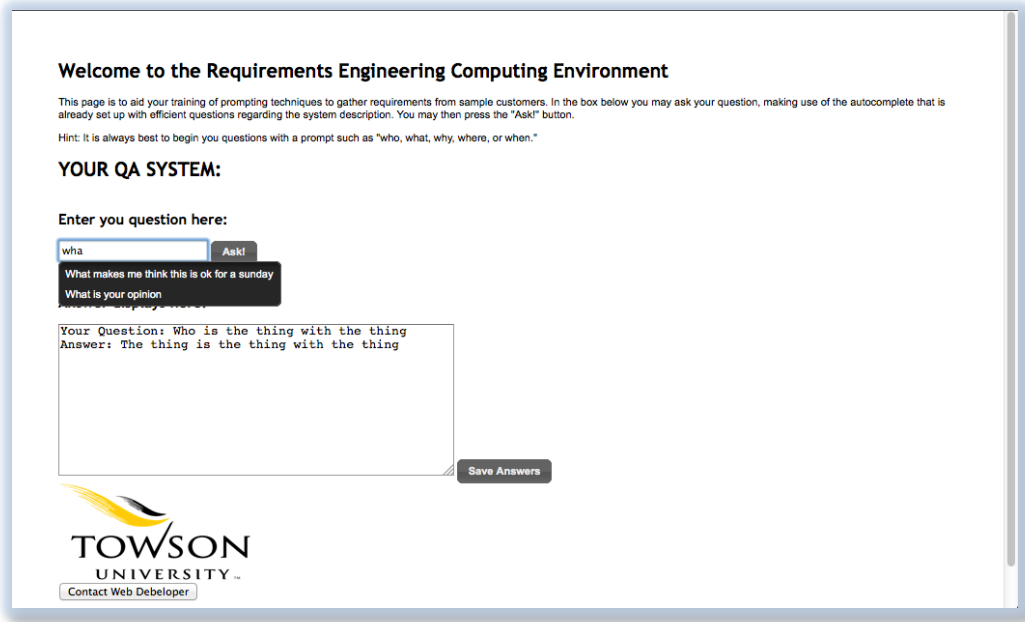


Figure 14 - Iteration 2 Final Front End

Additionally, the "Save Answers" button allowed the users to save their session, for analysis purposes (See Figure 18).

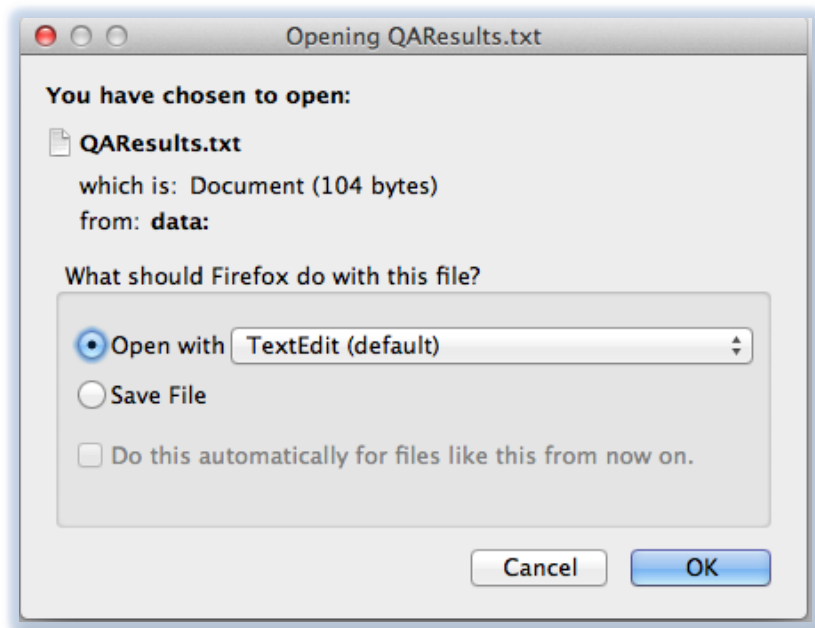


Figure 15 - Iteration 2 Save Dialog

Back End Development

The initial implementation of the chatbot was taking advantage of AIML and Pandorabot's chatbot hosting services. While AIML provided an XML structure for simple question-answer systems, there was an extreme amount of redundancy for a simple implementation of a chatbot. Additionally, the chatbot hosting site, Pandorabots, had proven to be reliable throughout the duration of this project.

The suggestion was made that data from keyed inputs from the users in conjunction with a look up table could quickly and easily appear to be a QA system. Implementation made use of the prepopulated keys serving as the prepopulated list of inputs, and use these as the keys for a collision free back end hashmap. While the initial implementation of this QA system would not have an enormous set of questions, future use may require an extensive set of inputs which will take advantage of the quick lookup capabilities of the hashmap. Javascript can successfully implement the hashmap at runtime preventing any need for storage of data on external servers.

Proposed Testing Results and Analysis

The pilot study was to determine the number of general, goal and process requirements acquired by each system user in addition to user interaction data being gathered. Due to time constraints on the development of this project, the pilot study on this iteration of the dialogue system was unable to be executed; however, the intentions of the proposed study will be discussed.

Development

Similarly to the second generation chatbot study, this study should to be performed by the graduate level Requirements Engineering students. Students should be encouraged but not required to participate in exchange for extra credit towards their final grade.

This study should contain a pre-task questionnaire and post-task survey as discussed in the pilot study research section of this document.

Participants should be given 15 minutes to interact with the dialogue system, taking advantage of the autocomplete functionality to ask as many questions and elicit the maximum number of requirements. Once satisfied with the number of acquired requirements, or having reached the time constraint, users should be asked to save copies of their transcripts and submit them to their Towson University Blackboard account.

Additionally, participants should be given a specification template with each of the requirement categories listed. Participants should be directed to fill out the template based on their interaction with the dialogue system; the completed specification template should then be submitted each student's Towson University Blackboard account.

Expected Results

The results from the pre-task questionnaire, conversation log, post-task survey and specification template should then be analyzed to correlate the types of questions asked with the number and type of requirements acquired. Additionally, there should be analysis performed on the most infrequently and frequently asked questions. Data gathered from the pre-task questionnaire will aid in determining what, if any, relationships there are between ethnic and gender backgrounds and selection of prompts.

The intention is that results of this study will contribute to further the training of future software requirement engineers. The questions a software engineer asks and the way that they follow up on those questions to probe clients for requirements provide information to aid in the requirements elicitation process and a system's development as a whole.

Conclusion

In an attempt to assist and further the requirements elicitation phase, we designed two interactive dialogue systems to model prospective client responses and respond in real-time to prospective developers' questions regarding a system's requirements. The first system employed an AIML based chatbot for answer questions created by the user, whereas the second system provided users with a listing of questions that would display as autocomplete to their input.

We developed a knowledge base for the interactive dialogue system by researching types of questions and their corresponding requirement categories to program potential responses. We took advantage of Browne and Rogich's semantic prompting technique to populate the knowledge base for the first iteration of dialogue system chatbot based on the fall 2013 Software Requirements Engineering syllabus.

Piloting the first iteration of system through integration with existing assignments as a part of a graduate Requirements Engineering course, we discovered that the first chatbot would not efficiently handle various types of input from students, which ultimately lead us to the development of the second iteration of dialogue systems.

Challenges and Limitations

Among the many challenges of this project, there were two major challenges and limitations manifested throughout the duration of this project.

The first challenge was the time constraint. This project was to be developed throughout a two-semester time frame. Initially, the interactive dialogue system was to have only two iterations, the pilot chatbot based on the Requirements Engineering syllabus, and the final chatbot based on a system description (KMC Auto Rentals) distributed to the Requirements Engineering class. However, upon discovering the major limitations of AIML, a second approach to a dialogue system had to be researched and implemented. By the end of the first semester, the first chatbot had been completed, leaving only a few months to develop an entirely new system along with knowledge base and question set creation. Once the system was implemented, there was no longer time to run the second study.

The second challenge to this project was the collection of limitations to the AIML language itself. During testing, the second generation of the chatbot performed exceptionally well with an 81% accuracy in responding to inputs. However, during the pilot study conducted in the class, the accuracy of responses fell to a 21.5%. For a detailed description of inputs failing to be handled by the chatbot, see section titled "Usability Study Results."

Future Direction and Developments

This project is an introduction to developing a much larger system aiding in the training of students pursuing and understand of requirements engineering and the requirements elicitation phase of the software development lifecycle. Additionally, the results and findings of this thesis will be used in a doctoral study expected to continue throughout the 2014 semesters.

Developments

Future developments should be the inclusion of an automated dialogue system within a much greater system to train users in requirements engineering as well as measure their level of success with the dialogue system. The anticipated system should include four categories

1. A tutorial on Requirements Engineering(RE)
 - a. General RE engineering overview including, but not limited to, the focus of RE, types of requirements, as well as work and business processes.
 - b. The different types of questioning techniques, such as the task characteristics, semantic, and syntactic prompting techniques (Browne & Rogich, 2001).
2. A timed interaction with an automated dialogue system
3. A specification template to query users of the requirements they gained as well as test their knowledge of requirement types. This would serve as the criteria for measuring how well each user performed.
4. A questionnaire to discover what each user learned as well as gain insight as to their questioning strategies.

Implementations

The intention is to have the new dialogue system as a part of a Massive Open Online Course(MOOC) course. The course would be available for students from various locations to train in the requirements elicitation phase (Cormier, 2008). Section 2 of the anticipated system would include an implemented dialogue system based on the second iteration discovered in this project. Development of the system would be conducted in two sections, the front end and back end developments.

Front End Developments

Front end developments should continue in the use of HTML, Javascript/jQuery, as well as Node.js. The Node platform will allow for a networked application without the need for an external web server. This will provide a lightweight solution for the client-database traffic (Dahl, 2007).

Back End Developments

The backend of the new system will implement Mongoose, a schema-driven NoSQL MongoDB database. This will provide an efficient way of creating objects that can be validated against a JSON schema to be stored in the NoSQL database. From Mongoose's website, a schema can be easily created and maps to the MongoDB as depicted below ("Mongoose," 2011).

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

var blogSchema = new Schema({
  title: String,
  author: String,
  body: String,
  comments: [{ body: String, date: Date }],
  date: { type: Date, default: Date.now },
  hidden: Boolean,
  meta: {
    votes: Number,
    favs: Number
  }
});
```

Table 8 - Mongoose Schema

For the implementation of the dialogue system, each pairing of question and answer will be stored in a schema that could include any additional information that might be useful for analysis. Additionally, interaction sessions between the user and the system can be tracked in a second schema to log the user, the duration of interaction, time of day, location of user via their IP address, as well as the results of their questionnaires, surveys, specification template and time spent reading directions and requirements engineering overview information pages. The extensive tracking of user sessions will provide an enormous amount of data for analysis of user interaction and system performance.

Appendices

Appendix A – Question Set for Prototype

1. Who is the instructor?
2. Who is the lab assistant?
3. Who is the teacher's assistant?
4. What is the instructor's contact information?
5. What is the course name?
6. What is the course ID?
7. What is the course description?
8. What is involved in the course?
9. When is the course offered?
10. What is the course schedule?
11. When is the final exam?
12. What are the project deadlines?
13. What are the course objectives?
14. What are the course prerequisites?
15. Who can take this course?
16. What assignments are in this course?
17. What projects are in this course?
18. What papers are in this course?
19. What exams are in this course?
20. What percentage of the grade are projects?
21. What percentage of the grade are papers?
22. What percentage of the grade are exams?
23. What percentage of the grade are assignments?
24. When does the class meet?
25. Where does the class meet?
26. How is the course graded?
27. What textbook is required?
28. What other suggested readings are there?
29. How can a student withdraw from the course?
30. When are assignments due?
31. When are the projects due?
32. When are the papers due?
33. When are the exams?
34. How are projects to be executed?
35. How are exams to be executed?
36. How are papers to be executed?

37. How are the assignments to be executed?
38. How are projects to be graded?
39. How are exams to be graded?
40. How are papers to be graded?
41. How are the assignments to be graded?
42. What is the purpose of the assignments?
43. What is the purpose of the projects?
44. What is the purpose of the exams?
45. What is the purpose of the papers?
46. Who is required to perform the assignments?
47. Who is required to perform the projects?
48. Who is required to perform the exams?
49. Who is required to perform the papers?
50. What outside help is there for the assignments?
51. What outside help is there for the projects?
52. What outside help is there for the papers?
53. What outside help is there for the exams?
54. What are the classroom policies?
55. When will the exams be graded?
56. When will the projects be graded?
57. When will the assignments be graded?
58. When will the papers be graded?
59. Where will the grades be stored?
60. Where is the course information?
61. How do students submit assignments?
62. How do students submit exams?
63. How do students submit projects?
64. How do students submit papers?
65. How do students receive graded exams?
66. How do students receive graded assignments?
67. How do students receive graded projects?
68. How do students receive graded papers?
69. Where can the textbook be found?
70. When will final grades be posted?
71. Where will final grades be posted?

Appendix B – Prototype Study Results

Prompt	Topic	Sub Topic	Missing Synonym	Missing Descriptor	Poor Ordering	Previous topic matched	Missing prompt	Incorrect Response	Irrelevant Question	Correct	Composite Prompts
When	course			Drop							
what	exams			weights							
what	paper		singular paper	weights							
what	mid-term		mid-term	weights							
how	assignments			many							
what	assignments			percentage							
when	class			last day of							
how	class			long							
what		required									
what	instructor's		instructor's	name							
what	textbook	required			required matched first						
which	(chapters)					course makeup	which				
how	course			taught				course makeup			
how	course		blackboard					course makeup			
what	(schedule)								x		
what	papers									x	
who	instructor									x	
what	instructor			name							
when	exam									x	
when	mid-term		mid-term								
what	research article		research, article							x	
what			research article								
what	presentation		presentation							x	
what	project		singular project								
what	assignments			grading						x	
what	assignments									x	
what		description								x	
	project		project				is				
what	exams			percentage/ worth				exam overview			
	homework		homework				is				
who	course									x	
what	course	prerequisites								x	
what			reading					course schedule			
what	course	name								x	
how	projects									x	
	projects						is				
what	components								x		
what	(software req eng)								x		
what	(learn)								x		
where	course			(available?)						x	
what/where	classroom								x	x	what/where
what	book		book	title							
why				(taught)					x		
how				(prepare)					x		
when	course			finish							
what	(your)	name							x		
when	homeworks	due	homeworks								
how	project		singular project	many							
when	course	objectives							x		
what	course	number		number			wrong prompt				
what	(holidays)								x		
what		office hours					wrong prompt				
what	(plagiarism)								x		
what	(cheating)								x		
what	course	policy									
what	(honesty)								x		
when	tests		tests								
what	lectures								x		
what	homeworks		homeworks								
what	course	policy/procedure s		policy/procedures							
	book						missing		x		
when	grade							x			
who	(you)								x		
when	(break)								x		
who	project			my group				x			
when	homeworks		homeworks	assigned							

Table 9 - Prototype Study Results

References

- Browne, G., & Rogich, M. (2001). An empirical investigation of user requirements elicitation: Comparing the effectiveness of prompting techniques. *Journal of Management Information Systems*, 17(4), 223-249.
- Cormier, D. (2008). *Mooc list*. Retrieved from <http://www.mooc-list.com/>.
- Dahl, R. (2007). *Node js*. Retrieved from <http://nodejs.org/>.
- Gasser, M. (2009). *The cognitive science of linguistics*. Retrieved from <http://www.indiana.edu/~hlw/>.
- Goh, O., Fung, C., & Depickere, A. (2007, June). *Domain metric knowledge model for embodied conversation agents*. 5th international conference on research, innovation & vision for the future(RIVF'07), Hanoi, Vietnam.
- Hart, D., & Goertzel, B. (2008). *Sentence algorithms*. Retrieved from http://wiki.opencog.org/w/Sentence_algorithms.
- Moldovan, D., Pasca, M., Harabagiu, S., & Surdeanu, M. (2003). Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2), 133-154.
- Mongoose*. (2011). Retrieved from <http://mongoosejs.com/index.html>.
- Pandorabots*. (2002). Retrieved from <http://www.pandorabots.com/botmaster/en/home>.
- Sentence Diagrammer*. (2009). Retrieved from <http://1aiway.com/>.
- Shawar, B. (2011). A Chatbot as a Natural Web Interface to Arabic Web QA. *International Journal Of Emerging Technologies In Learning*, 6(1), 37-43. doi:10.3991/ijet.v6i1.1502.
- Swedberg, K. (2013). *jquery*. Retrieved from <http://jquery.com/>.
- Wallace, R. (2003). *The Elements of AIML Style*.
- Wallace, R. "Reference AIML 2.0 Interpreter." *Program-AB*. ALICE AI Foundation, 21 Feb 2013. Web. 2 Dec 2013. <<https://code.google.com/p/program-ab/>>.
- Worswick, S. (2012). *Chatbot battles*. Retrieved from <http://www.chatbotbattles.com/>.

Curriculum Vita

NAME: Erika Marie Boquist

PERMANENT ADDRESS: 3600 Duxbury Court, Jarrettsville, MD 21084

DEGREE AND DATE TO BE CONFERRED: Master of Science., 2013

Secondary education: North Harford High School. Pylesville, MD. 2005

Collegiate Institutions Attended	Dates	Degree	Date of Degree
Towson University	2010-2013	Master of Science: Computer Science Concentration: Software Engineering	12/2013
Frostburg State University	2005-2010	Bachelor of Science: Music Concentration: Clarinet Performance Minor: Computer Science	06/2010

