A Novel Process Meta-model for Developing Automatic Speech Recognition Systems

Gabriel F. Araujo  
Federal University of Sergipe  
Computer Science Department  
Sergipe, BRA 49100-000  
Email: gabrielfa@dcomp.ufs.br

Hendrik T. Macedo  
Federal University of Sergipe  
Computer Science Department  
Sergipe, BRA 49100-000  
Email: hendrik@ufs.br

Abstract—Nowadays, Automatic Speech Recognition (ASR) figures as the main character in the Human-Computer Interaction (HCI) scenario, not only for its great usability but also for its recent advances. Several commercial applications have been launched with built-in speech interfaces. Yet, building a speech recognition system is not a trivial task. Although the speech recognition disciplines are very well documented, they occur in an unrelated way. It is difficult to understand the logic process for building such a system. We propose a development process meta-model for ASR systems in order to guide anyone to build their own ASR system. The process is validated with two distinct case studies: a context-sensitive ASR for navigating mobile robots and a semantic-driven ASR for controlling a home automation system.

I. INTRODUCTION

Automatic speech recognition (ASR) systems address the task of translating speech to text. It is a lot useful for interfacing humans with machines in their natural way of communication. Speech interfaces has been growing significantly in the past decade or so due the advances occurred in this research field [12]. Many types of commercial applications has successful incorporated such interfaces. Yet, the speech recognition problem is far from being solved. One of the biggest challenges for ASR system is to handle variability in speech, specially for speaker-independent tasks. Additionally, the ASR has a difficult job for handling environmental variability, such as channel noise and other external distortions [10].

In order to deal with all kind of issues, several techniques have been developed [12], [7]. Typically, each technique is better suited for specific issues and lacks the ability to handle others. As a consequence, the design of ASR systems becomes challenging when facing several variables. In the literature [7], [4], topics concerning each aspect of ASR are usually presented independently. Sequence and dependency relations are often implicit or hard to notice and understand.

So, how can we actually develop an ASR system from scratch? Is there a specific development process for guiding step-by-step how to achieve such a system? Most of the ASR systems are actually developed in ad-hoc way, for a specific purpose only [9], [13].

In this paper, we propose a development process-meta model in order to guide the ASR systems building. Such process has to be generic enough to cover as many aspects of the ASR as possible.

The rest of paper is organized as follows. We give a brief introduction to ASR systems and its basic architecture in section II. In section III we present the development process meta-model for ASR systems. We give an overview over the whole process and we detail each of its activities and tasks. Following, in the sections IV and V, we match the proposed model with two distinct case studies: a context-sensitive ASR for controlling the navigation of mobile robots and a semantic-driven ASR for controlling a home automation system. Finally, in section VI we conclude the work and discuss about limitations and future work.

II. ASR SYSTEMS

An ASR system aims at mapping an acoustic evidence to a sequence of words. Such sequence must be the most likely sentence that has been uttered in that evidence. In a preprocessing stage, the speech waveform evidence is appropriately represented into an acoustic feature vector $X = (x_1, x_2, \ldots, x_t)$ in order to be handled by computer. Therefore, an ASR system can be mathematically described as a mapping of such acoustic evidence $X$ to a sequence of words $W = (w_1, w_2, \ldots, w_n)$, as defined as follows

$$W = \arg \max_{W \in \omega} P(W|X) = \alpha P(X|W)P(W) \quad (1)$$

where the $\alpha$ is the constant factor $1/P(X)$ and the term $P(W|X)$ has been rewritten using Bayes’ Rule. Thus, the goal of the recognizer is to find the sequence of words that maximizes the underlying product. The evidence likelihood $P(X|W)$ is computed by an acoustic knowledge base, most known as acoustic model (AM). Similarly, the prior probability $P(W)$ is computed by a linguistic knowledge base, most known as language model (LM) [8]. The two compose the underlying knowledge base of the classic speech recognition architecture. Since the range of candidate sequence of words $W = (w_1, w_2, \ldots, w_n)$ that maximizes the product is huge, a search algorithm is used to perform an efficient search in the state space. This search process is known as the ASR decoding process.

Speech recognition systems can vary widely for distinct domains. From the most basic to highly complex and robust systems. They may require several others knowledge bases, scalability, real time response, extra processing stages, support
for many languages, adaptation, semantic analysis, etc. Therefore, the development process of an automatic speech recognition system for a generic application domain may be either simple or complex, depending on the nature of the recognition scenario. Yet, since these systems are mostly used as human-computer interfaces by several end-user applications, we have designed a meta-model to provide an end-to-end development process in order to guide any ASR system building task.

III. DEVELOPMENT PROCESS FOR ASR SYSTEMS

A. Eclipse Process Framework

The Eclipse Process Framework Composer or simply EPF\(^1\) is an open-source tool that aims at producing a customizable software process engineering framework and supports a broad variety of project types and development methods. Since most of these methods are documented in different ways, the EPF Composer has a predefined schema for allow you to describe and structure such methods in a standard way. This schema is based on the SPEM Specification (Software & Systems Process Engineering Meta-Model)[1].

The most fundamental principle in the EPF is the separation of reusable core method content from its application in processes. The method content describes what is to be produced, skills required, and the step-by-step to achieve the specific development goals. These method content descriptions are independent of a development life-cycle. Processes describe the development life-cycle. They take the method content elements and relate them into semi-ordered sequences that are customized to specific types of projects [6].

B. ASR system’s life-cycle

For being a component of a larger application, the development of an ASR interface must be done within a software development process. Here, we used the Open Unified Process or simply OpenUP, which is an open-simplified version of the RUP (Rational Unified Process) and part of the EPF project. Thus, the ASR development is performed among the elaboration and construction phases of the OpenUP model, as we can see in the figure 1. The former comprises the (i) Outline activity, whereas the latter is subdivided in two other activities: (ii) Modeling and (ii) Integration. In the following sections, we detail each one of these ASR-specific activities and their tasks.

C. Outline activity

This activity is enclosed in the elaboration phase, where we can identify the possible use cases and requirements of the speech interface. It has only one task: Specify the scope and domain. Such task addresses the scope of the speech interface, i.e., whether the system will be mono or multi-language and the type of the recognition task that will it support, such as isolated-word and large-vocabulary continuous speech recognition. It is also responsible for defining the vocabulary supported by the system, which is precisely linked to domain of the application.

\(^1\)https://www.eclipse.org/epf/

D. Modeling activity

This process comprises the activities necessary to build the resources of the ASR interface: the acoustic and linguistic models, as discussed in the section II, as well as other required knowledge bases. Yet, in order to build these resources, we also must to build intermediate ones, such as datasets (text and speech corpora), and the phonetic dictionary. The figure 2 depicts the activity diagram of the whole modeling activity. Each sub-activity in this process is responsible for building one of these resources and obeys their inter-dependency. The acoustic model is dependent on both speech corpus and phonetic dictionary; the language model is dependent on the text corpus; and the context and semantic models are independent of any resource. Next, we briefly describe each one of those sub-activities.

1) Build a speech corpus: It aims to build the speech corpus\(^2\), i.e., a collection of speech (audio) files along with its transcriptions. Thus, this activity produces the speech corpus deliverable, which consist of the sets of speech and transcription files. There are two tasks in order to accomplish this goal: record audio files and transcribe audio files, which must be performed in that order by the ASR developer role. In the former task, which outputs a speech

\(^2\)A corpus is a collection of written or spoken material in machine-readable form, assembled for the purpose of studying linguistic structures, frequencies, etc.
files set, we must systematically define a set of utterances and record them from a distinct group of speakers. The latter task is responsible for transcribing all the recorded audios in order to generate the transcription files set.

2) Build a text corpus: This activity is responsible for building the text corpus, i.e., a collection of written sentences which will be used for training the linguistic knowledge base. Two singles tasks summarize the efforts to accomplish this activity: collect texts and process texts, in that order. The first task aims to gather all the sentences that will make up this corpus. Such sentences can be collected in several ways, either manually or automatically, according to criteria of selection of texts and data sources (book, website, journal, etc). Following, the latter task address the processing of the texts previously collected, which might includes part-of-speech tagging\(^3\) and text categorization.

3) Build a phonetic dictionary: The goal of this activity is to build a phonetic dictionary that maps the phonemes for all of words that compose the target vocabulary, i.e., to specify which sequence of phonemes orally represents each specific word. Two tasks compose this activity: define the vocabulary and transcribe the phonetic. First we must identify and define the possible vocabulary of ASR users. Next, in the latter task, we must select a phonetic alphabet\(^4\) to represent the several phonemes and intonations of the language, and then, transcribe the phonetic of all of words in the vocabulary, i.e., define the sequence of phonemes that distinguish every word that may arise in the recognition task.

4) Model acoustic: This activity comprises building the acoustic model and extends the “Model knowledge” activity, which aims at building the knowledge statistic models and is composed by two single tasks: train model, which takes as input a dataset and generates a knowledge model; and perform evaluation, which takes as input any artifact (including a knowledge model) and generates its report evaluation. Such report will be useful for decision making concerning the performance of the evaluated artifact. Thus, this modeling process covers from the basic definitions and adjustments of the training task to the evaluation of the model. As in its parent activity, the process has a cycle that depends on the decision node suitable, as shown in the figure 3. Therefore, the report evaluation outcome will tell us whether the model is suitable for its purpose or not. If not, we can go back to the training process in order to adjust its parameters. If the model is already suitable, we are done.

As part of the model training task, we must define the model type (such as HMM, RNN, DNN, etc) and the linguistic units to be represented in such model. The latter is an important decision on the acoustic modeling, since word models are quite accurate for small-vocabulary tasks, but infeasible for large-vocabulary due the great amount of data required for training. Despite less accurate, phonetic models, on the other hand, are generalizable and do not require a large amount of data for training.

\(^3\)Part-of-speech tagging is the process of including syntactic and morphological information of language to words of a text corpus.

\(^4\)A phonetic alphabet is a notation to visually represent the several phonemes of the target language.

Fig. 3. Activity diagram of the model acoustic activity, which has been inherited from the model knowledge activity. It’s the most basic process that may be used to build any statistical knowledge model.

Fig. 4. Activity detail diagram of the model the language activity, which extends the model knowledge activity. For distinct models, the differences in the process are normally the input and/or output products of each task.

5) Model language: This activity concerns building the language model, which represents the syntax and, occasionally, the semantics of the language. As in the previous, it extends the model knowledge activity and inherits its tasks and process. The difference here stands for the tasks’ input and output, as shown in the figure 4. Again, the training task includes choosing the type of model that we will use, such as a context-free grammar, which are very restrictive and are more suitable for tasks with small vocabulary; or a n-gram model, which can avoid the efforts for building a broad coverage grammar, but fails in requiring a large amount of data for training. Therefore, they are more suitable for domain-independent applications and when enough training data is available.

6) Model context: This activity address the contextual model building, which is an additional resource to the classic ASR architecture as proposed in [3]. Such model may be applied in several ways for supporting the recognition process, for example, adapting the basic models (acoustic and linguistic). Indeed, the awareness of the environment that this model provides may become quite important for robustness of the system, specially for domain-dependent applications. A little different from the Model Knowledge activity, this one is composed by three tasks: select pertinent context, which is responsible for determine what context information will be relevant to the ASR at a given situation; engineer context model, which defines where and how such information will be extracted, managed and stored; and the perform evaluation task, as seen in previous modeling activities. Its activity diagram is shown in the figure 5.

7) Model semantics: The model semantics activity also aims at building an additional resource to the ASR: the semantic model. This model incorporates semantic knowledge
to the speech interface and is an unarguably important piece on the speech understand task, which is usually the next step after the speech recognition. This activity comprises the tasks define domain ontology, which is responsible for building a semantic database of concepts relevant to the recognition task; and the perform evaluation task.

E. Integration activity

With all the resources available, the next step in the process is to actually implement the ASR system. Thus, this activity has 3 goals: (i) assemble the speech recognition interface by connecting and setting up all of its components (which were have built in the previous activity); (ii) optionally implement an adaptation method; and (iii) validate the running ASR interface. Therefore, three tasks are responsible for accomplish these goals: Configure the ASR, Implement an adaptation method and Perform evaluation; as depicted in the activity diagram of the figure 6. The first aims to assemble the ASR interface and so, it outputs the ASR system. Usually, we must re-use a software engine tool that performs the decoding process for a given speech audio. There is several different implementations of the ASR decoder (or ASR engine) to handle various needs and limitations. We must choose the one that fits our speech recognition requirements. Next, we have to specify the engine parameters and set it up with the previously built knowledge models.

The second task in the integration activity is optional and addresses the inclusion of an adaptation method in the ASR system. Such adaptation might boost the robustness of the interface since it can adapt itself in order to increase its performance. First of all, we must define the target and source of adaptation. The target is which components of the ASR (models, input data, decoding algorithm) will be affected by the adaptation, i.e., its characteristics and/or content that will be changed. The source is from where the method is going to extract information in order to drive the adaptation. Next, since the source and target are defined, we can implement the method that will transform the target components with the source data. Moreover, we also have to define the adaptation triggers, i.e., the events that will initiate the adaptation process.

Finally, in the perform evaluation task, we can validate the whole ASR interface by measuring the performance and effectiveness of the built ASR, i.e., whether it meets the interface requirements requested by the system, such as accuracy, response time, noise tolerance, etc. Depending on this evaluation, we have the option to go back to previous tasks and re-adjust the engine, any of its components, or the adaptation method as well.

IV. Case study: Navigation of Mobile Robots

On this case study, the proposed process meta-model was used to redesign the development of a context-sensitive ASR interface for controlling the navigation of mobile robots[2], [3]. Thus, a development process based on our proposal was built targeting this specific ASR application. The final development process only differs from the base model in the modeling activity, as it is illustrated in the figure 7. Following, we detail how each task in the model was applied for this particular speech interface.

The outline activity covers the definition of the speech interface’s characteristics, which here stands for mono-language with supporting for Brazilian Portuguese. Yet, the interface must address a continuous speech recognition task, since the application requires handling non-constrained command sentences that might be modified to enhance the set of actions.

In the modeling activity, we have to adjust the base process model to fits ours needs and therefore, only the activities Build a text corpus, Model the language,
and Model the context are inherited. Since an acoustic model was re-used[11], the activities for modeling the acoustic, building a speech corpus, and building a phonetic dictionary were all skipped. From the optional knowledge bases, only the context model was built, since the application requires a context-sensitive ASR, and thus, the Model semantics activity was skipped as well.

Yet, all tasks in the Build a text corpus were replaced with a single task named auto-generate artifact. Since the application has an specific domain (mobile robot navigation), it is unlikely that we have a source to gather texts for building the corpus. Instead, it was automatically generated from a context-free grammar. The table I shows examples of sentences the user may speak. Three other corpus were generated by repeating the same task and by adding different amounts of general sentences extracted from the CETENFolha text corpus. Each of these corpora was used to build a 3-gram language model in the Model Language activity.

In the activity Model context, the arrangement of objects inside the current environment, i.e., the presence and visibility of such objects, was defined as the relevant context information for the application. These information were dynamically extracted from images captured by a digital camera, processed and then stored into a database.

A speech recognition engine was set up and connected to the built models in the task Configure the ASR of the Integration activity. Following the process as shown in the figure 6, it was defined a method that applies context information on the linguistic model adaptation, i.e., those information provided by the contextual model are enhanced and then used to adapt the language knowledge base, and therefore, the speech interface is capable of narrow its knowledge to the current robot’s environment. Finally, in the last task of the Integration activity, the built context-sensitive ASR was evaluated by the WIL (word Information Lost) metric. The lowest rates shown in results were up to 22.67% and 19.23% after the language model adaptation, for the most restricted language model and two different values of LM weight in decoding process.

V. CASE STUDY: HOME AUTOMATION

In this case study, the process meta-model is used for redesign the development process of a speech-based home automation system [5]. Such system must be able to recognize the uttered speech commands, interpret them, and trigger the requested device actions. Moreover, the application must be flexible enough for allowing the user to customize it, such as managing (adding, removing and naming) devices and home appliances. The adjustments made in the proposed process meta-model for this case study are depicted in the figure 8.

In the outline activity, a continuous speech recognition task supporting the Brazilian Portuguese language is defined for the application. Despite the continuous recognition, the uttered commands must comply a fixed structure defined by a context-free grammar. Similar as in the previous case study, in the modeling activity we have skipped the activities build a speech corpus and model acoustic, since we are re-using a acoustic model[11].

In the modeling activity, unlike the previous case study, a semantic model is built instead of a contextual model as the optional resource to incorporate into the ASR. This model aims to represent the semantic network of the concepts enclosed by the system and its information is used to auto-generate the text corpus and phonetic dictionary. Thus, the modeling process is redesigned to fit these product dependencies, as we can see in the figure 9 against the modeling activity diagram in the figure 2. Both of build a text corpus and build a phonetic dictionary activities have its tasks replaced by the auto-generate artifact, as in the previous case study. Following, the linguistic model is built upon the text corpus.

In the integration activity, the speech interface is set up and an adaptation method is implemented. Such adaptation occurs in the following way: the user update the semantic model, for example by adding, removing and naming devices; next, the text corpus and phonetic dictionary are auto-generated from the updated semantic model; and finally the linguistic model is re-built. Furthermore, the ASR is evaluated by the word error rate (WER), word information preserved (WIP) and

<table>
<thead>
<tr>
<th>ACTION COMMAND EXAMPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn to right.</td>
</tr>
<tr>
<td>Go 2 meters ahead.</td>
</tr>
<tr>
<td>Travel to the door.</td>
</tr>
<tr>
<td>Stop.</td>
</tr>
<tr>
<td>Walk.</td>
</tr>
<tr>
<td>Return 60 centimeters.</td>
</tr>
<tr>
<td>Spin to the right.</td>
</tr>
</tbody>
</table>
real time factor (xRT) measures. It was achieved a 23, 69% rate for WER and 72, 70% for WIP. The lowest xRT measured was 0.29 and achieved when limiting some search parameters on the recognition process.

VI. Conclusion

In this paper, we have proposed, presented and described a development process meta-model for ASR systems building. In the section II we gave a brief overview of ASR system's architecture, and next, in the section III, we presented the process model and detailed its main activities. In the sections IV and V, we matched the proposed process model into practical and distinct case studies for ASR interface building. The former comprises a context-sensitive ASR for navigating mobile robots, whereas the latter comprises a semantic-driven ASR for controlling a home automation system.

The proposed process meta-model is generic enough to cover distinct types of application domains. It is important for ensure its applicability and reuse. Further, the process may be extended to include some aspects not considered in this presented version. Despite generic, the process is still superficial, i.e., it mostly covers the elementary activities of an ASR development life-cycle.

Future works include a broader refinement of tasks and activities that composes the process, describing more deeply each stage of an ASR building. Yet, it may be extended to speech synthesis and understanding, in order to cover all spoken language processing areas and not only the speech recognition task.

Acknowledgment

The authors thank the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the financial support [Universal 14/2012, Processo 483437/2012-3] and Fundação de Apoio à Pesquisa e Inovação Tecnológica do Estado de Sergipe (FAPITEC-SE) for granting a scholarship to Gabriel F. Araujo.

References