European Clearing House
for Open Robotics Development
www.echord.info

ECHORD

Technical Report D2.1:

Robotic Voice Development Kit Design

E. Bastianelli*, G. Castellucci*, F. Giacomelli**, N. M. Manes**, L. Iocchi*, D. Nardi*, V. Perera*

Proposal full title: **SPEAKY for Robots**

Proposal Acronym: **S4R**

Name of the Coordinating Person: **Daniele Nardi**

Institution of the Coordinating Person (*): **Sapienza Università di Roma, Dipartimento di Ingegneria Informatica, Automatica e Gestionale**

Other participating institution (**): **Mediavoice S.r.l.**

Preface

SPEAKY for Robots (S4R) aims at fostering the definition and deployment of voice user interfaces (VUIs) in robotic applications where human-robot interaction is required. More in depth, S4R promotes speech technologies transfer towards manufacturing processes, to provide semi-automatic speech-based interface development for robotic platforms. This in turn will boost up the robot presence in manifold activities, by supporting a natural interaction with humans.

S4R specific goal is a novel **Robotic Voice Development Kit** (RVDK), namely a framework that supports robotic developers in designing voice user interfaces with little effort. RVDK is conceived as an interactive environment aiding designers to define the voice interface according to the desired application requirements; hence, it is adaptable to different application fields. In order to design and implement the RVDK, state of the art solutions about lexical vocabularies and knowledge representation to capture the semantics of the domain, and natural language processing technologies will be integrated to build the input for the speech processing system SPEAKY, that is currently commercialized by Mediavoice partner of the consortium.

S4R experiment targets two possible application domains for RVDK. The first scenario deals with **home services**, where a user controls a humanoid robot through a voice user interface within a domestic environment. The second scenario consists of an **outdoor robotic surveillance**, where the user is controlling the action of a wheeled robot capable of navigating in rough terrain**.** The goal of the experiments is to assess the performance of the voice interface implemented though RVDK. This project is thus a **joint enabling technology development** effort, whose scope falls within the **human-robot co-worker** scenario, addressing the **human-robot interfacing and safety** research focus.

In this report we describe the results of the activity carried out in Task 2 (description reported in the Appendix).

1 Introduction

In this report we describe the results of the activities of the project that are focussed on the design and definition of the architecture of the Speaky Platform integrating the voice communication system with the Robot environment and the design of the Robot Voice Development Kit (RVDK) that will be used by VUI developers. The RVDK defined in work package aims at extending the Speaky Development Environment of the Mediavoice Speaky Platform. The new RVDK will include a set of new functionalities for the implementation of voice user interfaces that are customized for specific robotic platforms and specific application domains.

Before addressing the specific activities developed within this task, it is worth recalling that the design of the RVDK builds on the previous analysis [TR11], which focussed on the main elements of the design process: the prototype implementations of simple VUIs for different robotic platforms, the construction of the grammars for the input specification of the ASR, the use of linguistic and domain knowledge and the definition and implementation of a set of support tools for the development environment and testing of the interface.

Another preliminary consideration is in order, concerning the ASR. After Loquendo has been acquired by Nuance, we have not been able to acquire additional licences for their English ASR engine. In order to get the process going we have decided to switch to the MS speech engine. Although this replacement was managed in a surprisingly short time frame, it required a substantial work. It also required to switch to a different standard language to express the grammars and, more generally, to a different interaction between the ASR and the system. However, this change opened up the possibility of integrating (as an alternative) MS SAPI, that can be used without any prior specification and may help resolving situations where the utterances of the user fall outside the scope of the input language defined for the application. This additional possibility will be considered in the future development of the project; a report on the introduction of the MS ASR will be provided in the next section on the system infrastructure.

The document is organized along the three main sub-tasks included in the work-plan. However, we anticipate the description of the VUI generation process and postpone the design of the wizard, since the latter heavily depends on the former. Consequently, the main activities required to design the Robotic Voice Development Tool are presented as follows:

- In Section 2 we describe the system infrastructure, focussing on the interaction with vocal input devices and with the robotic platforms, on the features of the Speaky SDK and finally reporting on the use of the new ASR.

- In Section 3 we present the overall design, describing the input and output of the process, the components to be designed, and the runtime support.

- In Section 4 we describe the design of the wizard that will support the generation process including all the steps described in the previous sections.

Finally, in order to provide some details of the overall grammar definition process, we include in appendix an example of the grammar and of the construction process involving the use of linguistic resources and the definition of frame structures that represent the specific knowledge to be used throughout the design process.

2 System infrastructure design

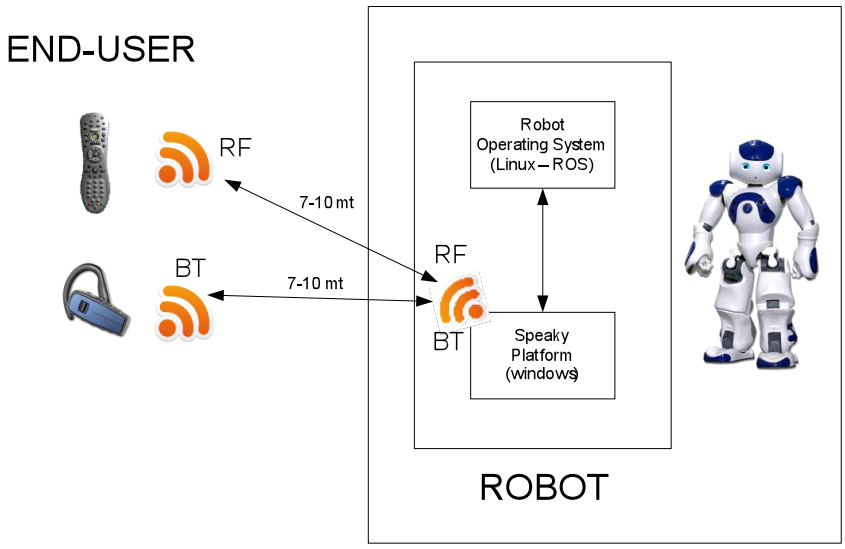
The first part of this report describes the overall architecture and the components of the Speaky SDK that are used to implement it. More specifically, this section describes the following:

1. The Speaky Architecture including the components to interface the robotic platforms, as well as the input devices.
2. The Speaky modules that support the development of S4R.

Indeed, Speaky has been developed to provide a suite of tools for speech interface development and S4R aims at exploiting it. On the other hand, during the process a number of additional components will be developed specifically to allow for an easy generation of dedicated interfaces for robots. Obviously such additional components will be afterwards available also for other applications developed through Speaky.

2.1 Speaky Architecture for S4R

The implementation is centered around Speaky components that are hosted on a dedicated computational unit and a client/server interface between the robotic platform with its on board computer. The Speaky Platform includes the Speaky Remote Control, through which the audio is acquired, and an application running Speaky software suite. The Audio input can be also acquired by other devices; the interaction with the input devices is handled by the underlying operating system. We have preliminarily examined off-the-shelf input devices to be used by the user for audio input. The new audio device must provide to be a good compromise between cost and accuracy and also give the user freedom of use and movement. We have considered many wireless devices (standard and proprietary); we then adopted a classic Bluetooth microphone or headset as depicted in the figure below. This new input device gives the user the possibility to speak having the freedom to use hands while performing other activities in the environment on which he/she is operating (i.e. indoor or outdoor environments). The user control for the activation of the device is not straightforward: previous experience by Mediavoice suggests that a manual control of the audio input improves significantly the performance of the speech engine.



As already mentioned, in order to facilitate the switch to different robotic platforms, we have kept Speaky on a dedicated host computer. The robotic platform and the Speaky PC are connected via client/server architecture on a TCP/IP connection. The PC running Speaky acts as a client sending commands to the robot as a result of

the speech recognition and the robot acts as a server waiting for commands to be executed. The overall configuration is shown in the above figure.

The above sketched architecture has been defined with the goal of reducing development time and focusing on the accuracy of the interpretation of vocal commands. Moreover, it achieves a very effective decoupling between the implementation of the interface and the robotic platform. This facilitates the deployment on existing robotic platforms, however, it requires an additional computing unit to be placed on the robotic platform or else kept in a location, where it can wirelessly communicate with the audio input device and with the robotic platform. Dedicated solutions that are based on the use of the onboard computing system of the robot must obviously take into account the computational and operating system requirements; they are outside the scope of the present project.

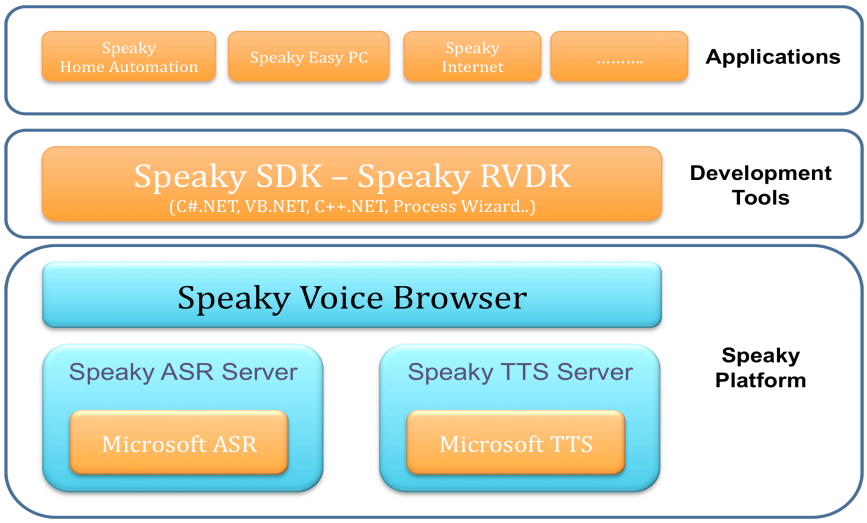
An additional benefit of the chosen architectural solution is the possibility to rely on the previous Speaky operating system configuration, based on MS Windows 7. The Speaky SDK is built on the Microsoft .Net Framework Technology and so can be integrated using any .Net programming language like C#, C++.Net or VB.Net.

In addition to ease of development, on MS Windows 7 we had the possibility to develop a new connection module for the Speaky ASR Server and Speaky TTS Server to replace the former tools by Loquendo with the new Microsoft Speech Platform (Text To Speech and Speech Recognition Engines). This new connection module, although unforeseen in the project proposal opened some new opportunity to our project and gave us some new tools to evaluate performance on the grammars developed for the experiment. In fact, the system can now more easily embed other speech engines including the MS SAPI or Google ASR.

Embedding new Microsoft Speech Platform in our architecture required a change in the formalism to express the grammar, so we switched from the ABNF to the XML syntax of the World Wide Web Consortium (W3C) standards for Speech Recognition and Speech Synthesis. More specifically, for Speech Recognition Grammars we use the SRGS/XML syntax [W3C, SRGS 2004] and for the Speech Synthesis we use the SSML (Speech Synthesis Markup Language) [W3C, SSML 2004]. Moreover, the meaning associated to the phrases or utterances defined for the user speech commands are defined using the W3C SISR (Semantic Interpretation for Speech Recognition) Specification [W3C, SISR 2006]. SISR tags of semantic interpretation can be added to speech recognition grammars to return to an application information on the recognition process, on the basis of rules and tokens that are matched by the speech recognizer.

2.2 Speaky Platform Modules

In this section we describe the Speaky modules, that we will integrate with the robot platform and that will be augmented with the new tools of the development Kit RVDK described in the next sections.



The Speaky Platform is built on three levels of software components. The Platform Level which is the core of Speaky includes three software modules, Speaky Voice Browser, Speaky TTS Server and Speaky ASR Server. The Speaky Voice Browser is the dialog manager of the platform: typically a dialog manager used in desktop applications, selects an interpretation based on the output of the Speech Recognition Engine, determines how the utterance fits into the dialog so far and decides what to do next. The Dialog Manager might need to retrieve information from an external source, or to forward information or data to an application and so on. If a response to the user is required, it will choose the words and phrases to be used in its response to the user, and transmit these to the Text-to-Speech System to vocalize the response to the user[1]. These tasks are executed by communicating with the application logic that in our case is the Robot. Feedbacks received by the Robot after the execution of the task can in turn be used to give a response to the user.

In order to better characterize the information flow among the Speaky core modules, it is worth focusing on notion of context used by the dialog manager. The context is used to characterize the state of the interaction with the user and, in the case of S4R, also the state of the robot. One specific feature of Speaky is the ability to activate the ASR by specifying the grammar at run time. A context is thus typically associated with one or more grammars, that can be used for the activation of the Recognition Engine in a specified state/context. The Voice Browser communicates with the Speech Recognition Engine through the Speaky ASR Server, activates and deactivates dialog contexts and sends the associated grammars for the recognition to the Speech Recognition Engine [Rabiner 1993] [Balentine at al. 2001]. The recognition process is thus driven (and restricted) by the grammar loaded in the current context.

The third component of the Platform Level of Speaky is the Speaky TTS Server, which manages the Speech Synthesis Engine [Dutoit 1997] [Allen 1991]. The dialog manager gives speech feedbacks to the end user sending text or text identifier to the Speaky TTS Server. Text or text identifiers are dynamically generated by the application logic based on the context of user Robot interaction.

The second level of Speaky includes the Development Tools. This component of the Speaky Platform is a set of application programming interfaces and a set of guidelines that support the developer during the definition and development of the Voice User Interface for a new or existing application [Weinschenk at al. 2000]. A Voice User Interface includes the definition of grammars, prompts and dialog flow diagrams. In Speaky speech enabled applications, the VUI is specified through simple xml files. This approach reduces the complexity of implementing ASR and TTS interface software to text editing operations. The application developer can thus concentrate on how to design the interaction, describing its steps, prompts, commands, on a higher and more dialog-oriented level. In S4R we can exploit the existing tool and methodology, while developing a suite of dedicated grammars and dialog components that can be used as a basis to customize specialized vocal interfaces for different platforms and application domains.

In any case, the design of the interface for a speech application should follow general usability principles briefly summarized below:

Feedbacks: The user needs feedback from the system to know what is going on during the interaction. When a user issues a speech command, the system should acknowledge the reception. Users also must be given feedback when the system is busy.

Confirmation: Confirmations are questions the system could ask to the user to resolve possible ambiguities on the dialog (i.e. ASR results are not showing a clear response). One needs to balance the cost of taking wrong branches in the dialog with the extra time and annoyance of the user for requesting confirmation.

User Expectation: Users expect the system to understand more than it is capable of, and to be able to provide more information than they can produce. The interface design can help the user to set appropriate expectations.

Prompting: Choose the appropriate words for your text. For example use the word “say” when you want the user to speak, rather than “enter”, use “enter” for inviting the user to press a key. Moreover, provide different ways of phrasing the same information/request to keep the user interested in the dialog.

[1] As an alternative the generation of the speech can be implemented using a TTS running on the robot.

Non-Speech Audio / Auditory Icons: Preserve standard sounds with their usual meanings (Siren: Emergency, Beep: Error or Attention and so on).

The upper level of the Speaky Platform is the application level, where the developer integrates the VUI defined for the application with Speaky through the Speaky SDK Application Programming Interface. In S4R we developed an application broker that includes the Speaky SDK and connects Speaky with the robotic platform.

In the next section we focus on the definition of the VUI, by addressing all the aspects that are specific to robotic applications. In the last section, we provide a detailed specification of the process for obtaining VUIs that are specific of a given platform and a given domain.

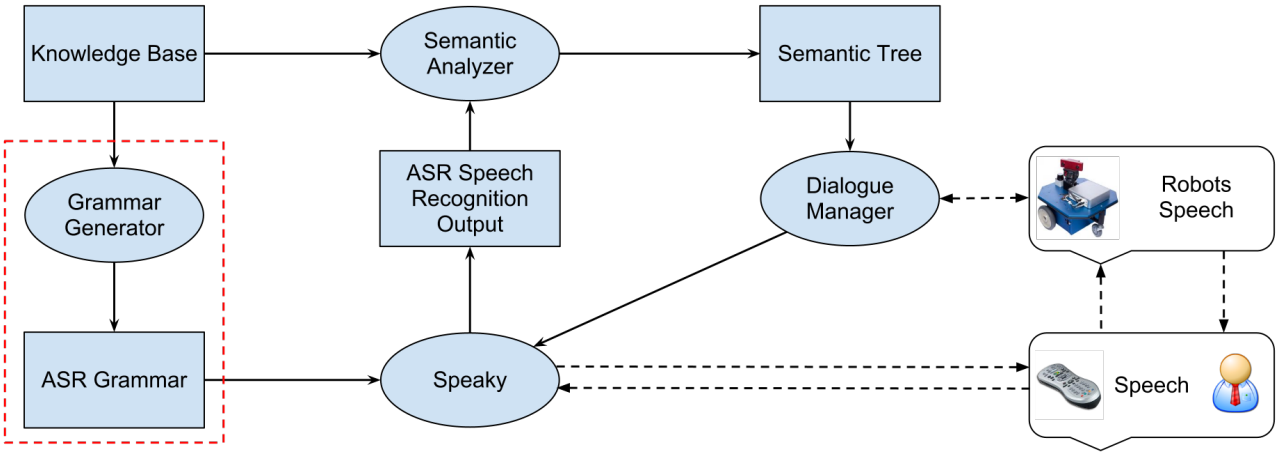
3 VUI Generation Process

The VUI generation process aims at providing all the components that are needed at run time to perform the recognition of spoken commands and the vocal interaction with the robotic platform. At the core of this process is the ASR embedded into the Speaky run-time, and thus the construction of the grammar to drive the ASR. However, the overall process must address several other components, some of them developed used existing tools in Speaky and others specifically designed for S4R. The figure below describes the information flow of the overall recognition process built around the ASR. The section of the diagram enclosed in a dashed line defines the off-line process of grammar construction, which is addressed in detail later on, while the other ovals represent modules that are executed at run time.

Speaky provides a component for the characterization of the meaning of the output of the ASR. However, in S4R we perform this function through a Semantic Analyser [Allen 1995] which relies on the domain characterization given in the knowledge base to determine the meaning of the result of the ASR. Such a knowledge base is also defined by the VUI generation process [Huang et al. 2001].

A dialog manager is embedded in Speaky and, as described in the previous section, it provides for a powerful characterization of the context of the dialog; however, since the interaction with a robotic platform requires a specialized approach to the design of the dialog manager, taking into account the state of the robot, also this component has been represented outside Speaky to emphasize the key role that it plays in the process. The definition of the Dialog Manager is another element of the VUI generation process.

The last component of the S4R runtime architecture, denoted as *Speaky*, includes all the other modules of the Speaky SDK that are not specifically modified by S4R, including the management of the input devices and the management of the interaction with the ASR (and with the TTS, when the Speaky TTS is used). In particular, Speaky manages the dynamic specification of the grammar for the ASR, the collection of the ASR multiple output, and possible multiple recognition requests for the ASR. While some of these functionalities have been implemented within S4R [TR11], they are not specific for S4R robotic applications and therefore they do not require specific input in the VUI generation process.



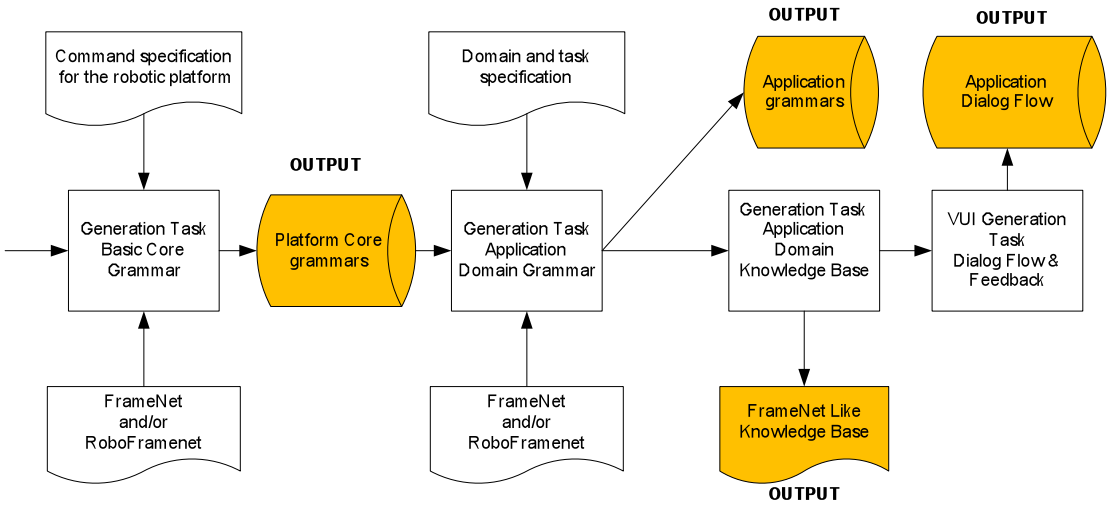
The approach proposed by S4R for the generation of the grammars that drive the recognition of the ASR is characterized by the development of a set of generic resources that can then be customized based on the specification of a given application. More specifically, in S4R we address two case studies that correspond to two types of robotic platforms and two types of application domains: humanoid and indoor wheeled robots and home and rescue application domains. Consequently, the generation of the VUI will rely on generic core grammars and generic application grammars. In the following, we will characterize them as: *Basic core grammars, Domain lexicon/grammar.* The process for building these grammars is needed for every new category of platforms and domains outside the case studies addressed in S4R. S4R will nonetheless produce a methodology and detailed guidelines to define new generic grammars.

The process for the generation of the generic grammars is illustrated in Appendix 1, using the NAO humanoid robot as a reference to build the generic core grammar for humanoid robots and the home domain. The generic grammars will be organized in small modular components in order to be easily reusable. In the definition of the generic grammars we rely on the linguistic resources, such as FrameNet or RoboFrameNet, to extend the basic language directly derived from the command specification, thus allowing the user to phrase the commands in flexible ways as described in [TR11].

As a result of the generic grammar construction also a generic *Knowledge Base* will be defined in the form of a set of frames. In particular, we will use a structure similar to the one used in RoboFrameNet, although we not necessarily aim at compatibility. The Knowledge Base is used by the semantic analyzer to match the meaning of the sentences that are recognized by the ASR. Such semantic analysis will be particularly useful to determine partial matchings, in cases where the ASR is not able to perform a complete recognition.

As explained in the previous section, the decision on which action to take, after all the relevant information has been attached to the candidate interpretations, is taken by the dialog manager. This component requires the specification of a set of states (contexts), and for each state: the grammar(s) to be used for the recognition of sentences, the actions to be done after an utterance of the user has been scrutined (both the actions of the robot and the feedback to the user) and the transitions to other states. It is important to notice that the state of the dialog will take into account also the state of the robot itself. Also in the case of the dialog manager, we arrange the VUI construction process by providing a generic definition of the structure of the dialog manager, that is based on the generic grammars previously described.

The construction of the generic resources, as above outlined, is a core element of the proposed approach, since it provides the basis for the customization process defined through the wizard, defined in the next section, where all the generic components will be targeted for a specific platform and application. In the following schema we describe the VUI generation process that generates the generic component.



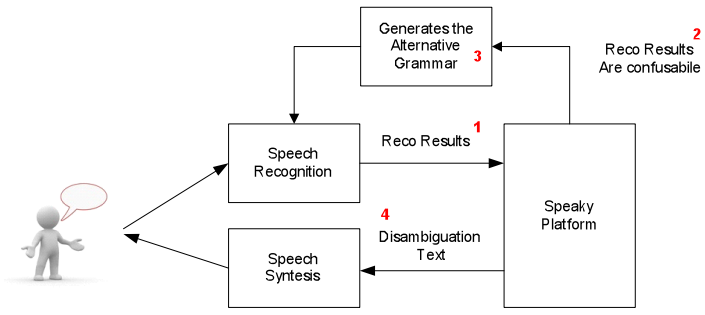
The resources provided for the wizard are those filled with the orange color and labeled with the “OUTPUT” string: platform core grammars, application grammars, knowledge base for semantic analyzer, dialog flow. The above process must be accomplished by the vendor of the S4R RVDK solution, any time a new category of platforms will be considered and any time a new application domain will be considered for an existing Robot platform.

Below, we establish the relationship between the development supported by Speaky and the new approach developed by S4R. The grammar construction process is a key and challenging task, when building a VUI for a given application and thus Speaky SDK specifically addresses it. The difficulties arise mostly in determining whether a grammar satisfies the needs of the application; tuning grammars is also needed to improve the response of the application to voice input. Thus, the main steps that Speaky guidelines define for grammar development include the following:

- Design and Authoring
- Testing and Debugging
- Tuning with Utterances
- Deployment

While the above steps are supported by the Speaky SDK, in S4R we have developed a dedicated approach for the grammar construction. The grammar specification in S4R is based on a syntactic characterization of the natural language sentences corresponding to the commands of the robot and on a subsequent characterization of the semantic elements that are admissible within these sentences. This process, which is supported by the linguistic resources FrameNet and RoboFrameNet, has been preliminarily described in [TR11] and is illustrated in detail in the Appendix.

The other key element in the design of a VUI for a given application is the response of the application to voice input. Since, many words and phrases are phonetically similar and the recognition of the ASR often produces several possible results with similar confidence values, the Speech Platform should try to disambiguate and recover errors in the recognition process. The disambiguation strategy currently adopted in Speaky is based on a new step of interaction with the user generating a run-time grammar, which contains only the voice commands to disambiguate. When the results of a recognition process have the same confidence score and a different semantic interpretation Speaky generates a grammar on the fly and prompts the user to choose between the phrases involved in the disambiguation. Of course we augmented each voice command with new alternative phrases in order to avoid or minimize further disambiguation steps. For example the first phrase is also augmented with the phrase “option one”, the second phrase with the phrase “option two” and so on.



In S4R we propose to extend the approach to disambiguation in two ways: first the component called Semantic Analyzer will try to provide possible semantic interpretation when the result of the ASR is partial. Moreover, the dialog manager will implement a decision procedure that is based on a richer set of inputs, including semantic interpretations, multiple analysis of the same input and state of the robot, and, possibly, on machine learning techniques.

A detailed description of the customization process to be implemented through the wizard will be provided in the next section.

4 Analysis and Design of the Wizard

The final part of this report focuses on the RVDK wizard, whose purpose is to allow for the customization of the VUI for specific robotic platform and applications. Before defining the wizard, we recall that the activities done for the VUI generation process are divided in two main phases that we call Resource generation phase and Wizard phase. The Resource generation phase, described in the previous section, builds the generic components, to be used during the wizard setup based on the choices made by the developer. The activities of the first phase are executed only once by the producer/seller of the Robot/RVDK and are not repeted by the developer during the execution steps of the wizard. The Wizard phase of the VUI generation process involves the activities of customization of the grammar domain, the knowledge base and the management of the disambiguation process. The Wizard activities instead are executed by the developers any time that a new end user or a new end user class needs to apply a specific platform for her/his personal application domain.

Below we describe in detail the input (in addition to the generic components), the output of the VUI generation process and the generation process itself.

Input

Command specification for the robotic platform: this set of commands are closely related to the specific robot in fact they refer to the low level controls of the system such as actuating joints or turning on and of specific sensors. This specification will be used to refine the generic core grammar.

Domain and task specification: this specification provides information about the environment the robot will be deployed in and the task it is supposed to carry on. This specification includes high level commands that depend on the domain and task and will be used to refine the generic application grammar.

Output

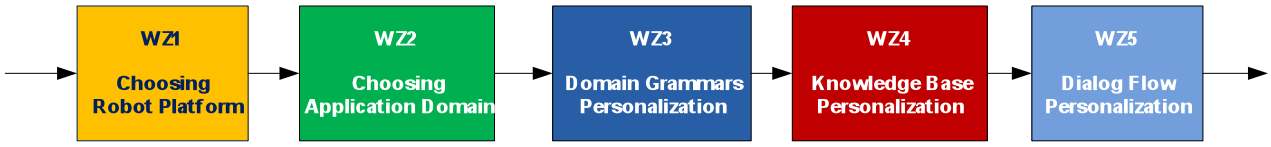
Platform specific core grammars: these grammars will enable the ASR engine to recognize all the sentences related to the capabilities of the robot with no connection to the environment or task.

Application specific grammars: this component can be seen as the dual of the previous one, in fact these grammars will enable the ASR engine to recognize all the sentences related to the environment in which the robot will be deployed and to the task it is supposed to carry out.

Knowledge Base for semantic analyzer: this component is a specialization of the generic knowledge base that is specific to the grammars produced in the previous steps.

Dialog flow: this component is a specialization of the structure used by the dialog manager, which is used to choose the action to be performed by the robot and the feedback to be provided to the user; in particular, the dialog flow will provide a refined specification of the states, including priorities on the possible actions to be executed in the state and a specification of the feedback for the user.

Now that the input and output of the VUI generation process are defined, we can describe how the generation process is arranged. The wizard that will enable the customization of the VUI will include the steps shown in the following schema.



The first step of the development wizard (**WZ1**) is choosing the robot platform to use, that is for example using a humanoid robot or a wheeled robot;

The second step (**WZ2**) instead consists of choosing the application domain where the robot will be used, for example the Home service domain or the outdoor- surveillance;

These two steps will add to the wizard process the appropriate platform and application grammars to be personalized in the next steps of the wizard, that is “Platform Specific Core Grammars” and “Application Domain Specific Core Grammars” as described in the previous section.

The third step (**WZ3**) consists of the generation of a specific grammar for the platform in use and the given domain/task. This step will be accomplished starting from the command specification of the platform, the basic core grammars, the domain lexicon/grammar and the available linguistic resources. This process is further structured into two steps, the first one concerning the refinement according to the robot capabilities and the second one referring to the specification of the instance of environment at hand. The customization of the grammar involves the addition/deletion of grammar chunks, phrases, names, objects and specific words depending on the specific application requirements.

The fourth step of the wizard (**WZ4**) is the personalization of the Generic Knowledge Base created as a data repository for the wizard and described in the previous section. In this task the developer acts like in the previous step and so adds or removes away frames in the Knowledge base depending on the specific domain on which the robot will be used and particularly the personal end user domain.

The fifth step of the wizard (**WZ5**) is the personalization of the Dialog Flow created as a generic Dialog Flow by the process defined in the previous section. In this task the developer adds or removes dialog states, define new feedback for the end user, personalizes predefined feedbacks and even change the actions to be performed by the robot after a particular utterance has been recognized. As stated in the previous steps all these activities depend on the specific domain in which the Robot will be used and particularly the personal end user domain.

In addition, to the above listed steps the development of the wizard will also address the following:

1. Analysis of the most effective technologies for the development of the wizard;
2. Design of the wizard, including the detailed specification of all the steps of the process;
3. Implementation of the wizard process.

References

[Huang et al. 2001]	X. Huang, A. Acero, H.-W. Hon, R. Reddy, “Spoken Language Processing: A Guide to Theory, Algorithm and System Development”, 980 pag.; Prentice Hall; ISBN: 0130226165; 1st edition (2001).
[Rabiner 1993]	L. Rabiner, “Fundamentals of Speech Recognition”, 496 pag.; Prentice Hall; ISBN: 0130151572; 1st edition (1993).
[Allen 1995]	J. Allen, “Natural Language Understanding (2nd Edition)”; Addison-Wesley Pub Co; ISBN: 0805303340; 2nd edition (1995)
[Dutoit 1997]	T. Dutoit, “An Introduction to Text-To-Speech Synthesis (Text, Speech and Language Technology, V. 3)", Kluwer Academic Publishers; ISBN: 0792344987; (1997).
[Allen 1991]	J. Allen, "Overview of Text-to-speech systems" , in Furui, Sondhi "Advances in Speech Signal Processing", Marcel Dekker; ISBN: 0824785401; (1991).
[Balentine at al. 2001]	B. Balentine, D. P. Morgan, W. S. Meisel, “ How to Build a Speech Recognition Application: Second Edition: A Style Guide for Telephony Dialogs“, 414 pag.; Enterprise Integration Group; ISBN: 0967127823; 2nd edition (2001).
[Weinschenk at al. 2000]	S. Weinschenk, D. T. Barker, “ Designing Effective Speech Interfaces”, 406 pag.; John Wiley & Sons; ISBN: 0471375454; 1st edition (2000).
[TR11]	E. Bastianelli, G. Castellucci, F. Giacomelli, N. M. Manes, L. Iocchi, D. Nardi, V. Perera, “Robotic Domain Definition and Representation”, Echord S4R Technical Report 1.1, February 2012.
[Framenet]	Framenet. URL https://framenet.icsi.berkeley.edu/fndrupal/home .
[W3C, SRGS 2004]	W3C. Speech recognition grammar specification version 1.0 w3c recommendation, March 2004. URL http://www.w3.org/TR/2004/REC-speech-grammar-20040316/
[W3C, SISR 2006]	W3C. Semantic interpretation for speech recognition (sisr) version 1.0 candidate recommendation, January 2006. URL http://www.w3.org/TR/2006/CR-semantic-interpretation-20060111/ .
[W3C, SSML 2004]	W3C Speech Synthesis Markup Language (SSML) Version 1.0 W3C Recommendation 7 September 2004 http://www.w3.org/TR/speech-synthesis/

Task 2: Robotic Voice Development Kit Design [M 2-6]

Participant	Role	Person-months
MV	Leader	8
UR1	Participant	3

Objectives:

This task describes the main activities required to design the Robotic Voice Development Tool:

- Identification of possible communication and data exchange protocols (HTTP, WIFI, ...) for the interaction of the two components, Speaky and Robots, and parameters to be communicated from one system to another. Speaky intercepts the voice command and notifies it at the system that controls the robot and then, the robot will execute the command, and eventually, returns a result to Speaky, who shall communicate to the user.

- Identification of the basic processes for creating a wizard, to be used by a robotic user, using the Robotic Voice Development Tool for developing the voice interface. For example, the user can choose the type of robot to be used (humanoid or tracks), the type of environment (home, stage, garden). Essentially this activity aims to identify all steps that are necessary for the user to achieve its specific voice interface.

- Specification of a voice interface generation module, which is integrated in the wizard and supports the generation of the speech language, that constitutes the input of the speech recognition module. This module will thus exploit the vocabulary, knowledge and linguistic constructs that are produced by Task T1.

The output of this task will support the activities conducted in T3. In fact, the output includes the diagram of processes, that defines the order and manner in which these processes must be developed.

Description of work, and role of participants:

T2.1 System Infrastructure Design
Being the goal the integration between the two components, in the first part of the task MV and UR1 will work closely.
The work will be split up in different stages:

- Sharing technological components to be used to handle vocabulary, domain knowledge representation, and robot control interface.

- Compilation of the entire list of the keywords to be used for the different components.

- Analysis of the state-of-the-art of wireless communication technologies, in order to choose the best way to transfer data from one system to the other. Particularly, the following features need to be taken into account within the capacity planning: Distance between the user and the robot; Data rate; Size of data to be transferred; System responsiveness.

- Design of the final architecture, which includes: Speaky, the infrastructure system, and robotic systems.

T2.2. Wizard Design
MV and UR1 will initially work together, in order to identify the voice interface developer's design goals and the range of choices allowed to him.

The work will be split up in different stages:

- Analysis of the choices needed by the voice interface developer in order to implement his voice interface. For example, the voice interface developer needs to choose the kind of robot to be used (humanoid or tracked) or the kind of environment which the robot will move in (open or closed) etc;
- Compilation of the entire list of the automatically allowed and forbidden choices, depending on the already made choices, and the possible graphic display order of such choices;
- Analysis of the processes arising from the choices made and of the relative actions to be undertaken. For example, choosing a tracked robot will require the addition of instructions to the grammar such as “move”, while choosing a humanoid one will require instructions such as “walk”;
- Analysis of the most effective technologies for the development of the wizard, which will need to be easy to use and to understand, and for the implementation of the whole voice environment that will arise from the wizard and design of all the steps necessary for the development of the wizard.

T 2.3: Voice Interface Generation

In this third part the voice interface generation process will be finalized jointly by UR1 and MV.

The work will be split up in different stages:

- Analysis of the results of T1 about the vocabulary and knowledge to extract the set of terms to be included in the language labeled with their grammatical role. Particular care will be needed in defining the process for identifying synonyms considering both the vocabulary and the domain ontology.
- Analysis of the results of T1 about the grammatical constructs to be allowed in the language. The size of the resulting set of phrases needs to be carefully controlled in order to ensure the performance of the speech recognition process
- Specification of the integration of the interface generation in the wizard and design of all the steps necessary for the development of the voice interface generation process.

A.1 Grammar Generation Process

In this appendix we describe in detail the process used to build the grammar for the Voice User Interface. In particular, we use as an example the NAO humanoid robot by Aldebaran Robotics to be deployed in a domestic environment.

This appendix is structured in 3 sections, one for each of the steps envisaged to generate the grammar: in Section 1.1, we address the robotic platform and describe its commands and behaviors; in Section 1.2 we show the basic syntactic structure of the grammar, customized based on the linguistic structures required by the language for the robot commands; in Section 1.3, we extend the grammar in order to account core behaviors of the robot and the semantics for the domain and application chosen.

A.1.1 Commands and behaviors of the robotic platform

Our first goal was to find a set of low level commands that the robot is able to perform; we therefore looked at the actuators and sensors of the humanoid robot. We initially refer to the NAO robot, which has a very rich set of sensors and actuators. For a detailed description see: http://www.aldebaran-robotics.com. However, in order to allow for the generalization of the process and make it applicable to other humanoid platforms, the specification is not restricted to the features and behaviours available on the NAO, and it will be reconsidered in the next steps of the project.

We start by considering a small set of commands referring to the simplest actions performed by the robot, without taking into account the feedback from the sensors. For each command, we provide an abstract syntax (which should be specified in the concrete language of each platform), a description, characterizing the effect of the command, preconditions (to be taken into account into the dialog) and prerequisites (to be checked when instantiating the language for a specific platform through the wizard). The basic commands are shown in the following table:

COMMAND	DESCRIPTION	PRECONDITIONS	PREREQUISITES
TURN_OFF()	Turn off the robot		
SNAPSHOT()	Shoot a photo		Camera
VIDEO_ON() VIDEO_OFF()	Turn on or off the camera(s) of the robot	Video OFF/ON	Camera
SAY(X)	Say the string specified by X (using the TTS)		Speakers
PLAY(X)	Play the sound specified by X (audio file)		Speakers
CONNECT(X)	Connect to the network specified by X (WiFi, Bluetooth…)		Connection Network
MOVE_RFOOT_FW(X) MOVE_LFOOT_FW(X)	Move the left or right foot in the forward direction of X cm, X can be both positive or negative (hip control)		Legs
MOVE_LFOOT_L(X) MOVE_RFOOT_L(X)	Move the left or right foot in lateral direction of X cm, X can be both positive or negative (hip control)		Legs
LIFT_LFOOT(X) LIFT_RFOOT(X)	Lift the left or right foot of X cm (leg control)		Legs
LOWER_LFOOT(X) LOWER_RFOOT(X)	Lower the left or right foot of X cm (leg control)	Foot L/R lifted	Legs

LIFT_L(X) LIFT_R(X)	Lift the left or right arm of X degrees, if X<0 the arms are lowered (shoulder pitch control)		Arms
BEND_L(X) BEND_R(X)	Bend the left or right elbow of X degrees (elbow control)		Arms
ARM_SIDE_UP_L(X) ARM_SIDE_UP_R(X)	Raise sideways the left or right arm of X degrees, if X<0 they are closed (shoulder roll control)		Arms
RAISE_HEAD(X)	Raise the head of X degrees (head pitch control)		Head
ROLL_HEAD(X)	Roll the head of X degrees (head roll control)		Head[1]
TURN_HEAD(X)	Turn the head of X degrees (head yaw control)		Head

[1] Not available on NAO

Next, we expand this set of commands with basic behaviours; these behaviours are typically obtained through a sequence of the simple commands described above.

COMMAND	DESCRIPTION	PRECONDITIONS	PREREQUISITES
SHAKE_HEAD()	Shake the head, as if saying no		Head
NOD_HEAD()	Nod the head, as if saying no		Head
KICK_L() KICK_R()	Kick using the left or right foot	Standing	Legs
STAND()	Stand up	Not standing	Legs
SIT()	Sit down	Standing	Legs
PUSH()	Lifts the robot's arms and starts walking as if pushing something in front of it.	Standing	Arms, Legs
WAVE_L() WAVE_R()	Wave the left or right arm		Arms
WALK(X)	Walk for X cm	Standing	Legs
TURN(X)	Turn of X degrees, if X>0 it turns clockwise	Standing	
CLAP(X)	Clap the hands X times		Arms
DANCE(X)	Dance for X seconds	Standing	Legs
BOW(X)	Bow, X specifies the degree of the hips joint	Standing	Legs
POINT(X,Y,Z)	Turns the head in the direction specified by X,Y,Z		Head

Next, we consider the behaviours involving a sensor feedback, by first addressing object handling (without localization).

COMMAND	DESCRIPTION	PRECONDITIONS	PREREQUISITES
GRAB(X)	Take the object X	Hands free, object X available	Grab Objects [1]

DROP()	Release any object held	Holding object	Grab Objects [2]
OPEN (X)	Open object X	Object X closed	Open Objects [3]
CLOSE (X)	Close object X	Object X opened	Close Objects [3]
LOOK_AT(X)	Take photo/video of object X	Object X available	Detect Objects
ON(S) OFF(S)	Turn ON/OFF the switch S	Switch S available, S OFF/ON,	Operate Switches [4]
ON_CONNECT(X) OFF_CONNECT(X)	Connect to device X (i.e. a Bluetooth enabled television)	Device X available	Connection to X
FOLLOW(PE)	Follow the person PE	Person PE available	People Detection and Tracking

[1] NAO simply positions the arms so that an object can be held
[2] NAO just restores its arms in the original position
[3] NAO is able to open/close objects by pushing
[4] NAO is able to turn on/off switches by pushing

Next, we consider complex motion, which can refer to a position relative to the robot (relative motion), or an absolute position, which requires global with localization.

COMMAND	DESCRIPTION	PRECONDITIONS	PREREQUISITES
GOTO (P)	Go to position P, expressed in the robot reference frame		Navigation
GOTO_POS(P)	Go to position P, expressed in a global reference frame		Localization, Navigation [1]

[1] Not available on NAO

If the robot is also localized in the environment several behaviors become possible, corresponding to actual tasks that the robot can perform. Correspondingly, the language requires to handle the locations in the environment that can be used to characterize the elements of the environment that can be dealt with by the robot. In particular, many (static) objects in the environment have a location, that can be referred to implicitely, when making a request to the robot. The location is denoted as an explicit parameter for the command.

COMMAND	DESCRIPTION	PRECONDITIONS	PREREQUISITES
GOTO (L)	Go to location L, expressed in the robot reference frame	Location L available	Navigation
GOTO_LOC(L)	Go to location L, expressed in the word reference frame		Localization, Navigation [1]
BRING(O,L)	Carry object O to location L and drop it	Holding object O	Carry Objects*
CHECK(O,L)	Go to location L (optional parameter) and take photo/video of object O		Detect Object*
FIND(O,L)	Find the object O in location L		Detect Object*
GO_OUT()	Exit the current location		*
ENTER(L)	Enter location L		*
TELL(X,PE)	Says the phrase specified by the sentence X to person PE.		Person Detection**

* requires navigation and localization;
** if location of person PE is available localization may not be required.

A.1.2 Syntactic structure of the grammar specification

Starting from the NAO core commands defined above, a SRGS grammar has been defined as described below. First of all, a semantic frame has been selected for every core command, according to the FrameNet description. The selected frame, with all its frame arguments, represents the action expressed by the command itself so; for example, the command:

WALK(X)

expresses the action of moving for X centimetres. The frame associated with this kind of action is SELF_MOTION. The DISTANCE frame element of this frame can then represent the number of centimetres to be walked, as well as the SELF_MOVER frame element refers to the entity that moves (the robot itself).

Frame: SELF_MOTION
Frame Elements:
DISTANCE: X
SELF_MOVER: Robot (implied)
DIRECTION: not instantiated
MANNER: not instantiated

Using this approach is possible to map each frame into a command and each frame element into a command argument, as above described. So, given the frame with its instantiated frame elements, the correct robot command can be identified.

Once the mapping between frames and commands has been defined, the most representative lexical units have been selected for every frame starting from the frame lexical unit lists given in FrameNet. So for example, according to FrameNet, the frame SELF_MOTION can be evoked by a large number of verbs. In order to generate a first manageable grammar, just verbs as "walk" and "advance" have been chosen, while verbs like "trek" or "crawl" have been eliminated. The frame template structure has been enriched with this information, as shown below:

Frame: SELF_MOTION
Frame Elements:
DISTANCE: X
SELF_MOVER: Robot (implied)
DIRECTION: not instantiated
MANNER: not instantiated
Lexical Units:
walk, advance

Given the different frames with their LUs, a "mini"-corpus of imperative plain sentences has been generated. This corpus has been populated with representative command sentences obtained using the LUs as main verb and considering possible commands for the robot. For example, for the frame SELF_MOTION, sentences like

"Walk three meters to the right"
"Advance forty centimetres"
...

have been added. The aim of this step is to study the different syntactic manifestation of the commands in the natural language form.

From the corpus obtained in the previous step, the grammar has been generated taking one command at time and studying its syntactic structure in terms of categories such as Verb Phrase (VP), Noun Phrase (NP), Prepositional Phrase (PP) and so on. The grammar has been built incrementally, adding or modifying production rules, in order to obtain the most compact grammar as possible covering all the commands in the corpus. Finally, all the words composing the sentences have been added as terminal symbols of the grammar, divided by their morpho-syntactic category (i.e. verbs, noun, preposition, adverbs ...). In this way, we capture the basic syntactic structure of the sentences, as characterized by the set of robot commands.

The grammar has been developed in a separated environment to test its consistency and correctness (i.e. Prolog), and then translated in the XML-SRGS language for the ASR engine.

A.1.3 Semantics of robot core behavior

Once the overall syntactic structure of the grammar has been outlined, the problem of embedding the semantic in it has been analyzed. First of all, a distinction between the *semantics of the robot* and the *semantic of the specific application* has been made in terms of different frames involved. The two different cases are described in the section A.1.3.1 and A.1.3.2. The semantic discrimination realized at this step of the development concerned just the grammar terminal symbols (i.e. the lexicon), with the aim to subcategorize words according to their possible semantic function. Words like adverbs have been analyzed and categorized out of a particular case, because they represent common frame elements among the different frames (i.e. the **non-core** [Framenet] frame elements like MANNER or DEGREE). The adverb syntactic category (Adv) has been divided in three subcategories, so that one adverb can be one of the following:

degree: a bit, a lot, a little, widely...
manner: quickly, slowly, softly, gently...
directional_modifier: out, down, right, left, forward...

This sub-categorization reflects the correspondence between an adverb and its semantic function, that is the evoked frame element. Using the semantic <tag> is then possible to let the grammar returns the value "degree: a bit" according to the SISR specification. For the subcategory "directional_modifier" it is necessary to add extra semantic information to each adverbial utterance, because the semantic function played by these is more accurate and not as general as an universal directional modifier. In fact, a directional adverb can specify the source, the destination or the direction of a movement: these adverbs play the role of specifiers with respect to frame elements such as GOAL, SOURCE or DIRECTION. So, a semantic label has been added to the respective adverb, as GOAL to "to", SOURCE to "from" or "out", as DIRECTION to "right" or "left", to specify the functions with respect to a noun labeled as generic LOCATION. As an example we can look at the following sentence:

“Get out of the kitchen”

the adverb “out” following the verb “get” specifies two fundamental things: the first, that the action is about going out, enriching the meaning of the verb; the second thing, linked to the first one, is the specification of the frame element composed by the location “kitchen” as a SOURCE frame element. Labeling the adverb “out” with the SOURCE label, we can say that the first LOCATION element after this adverb represent the SOURCE frame element of a frame like SELF_MOTION.

A.1.3.1 Semantics of robot core behavior

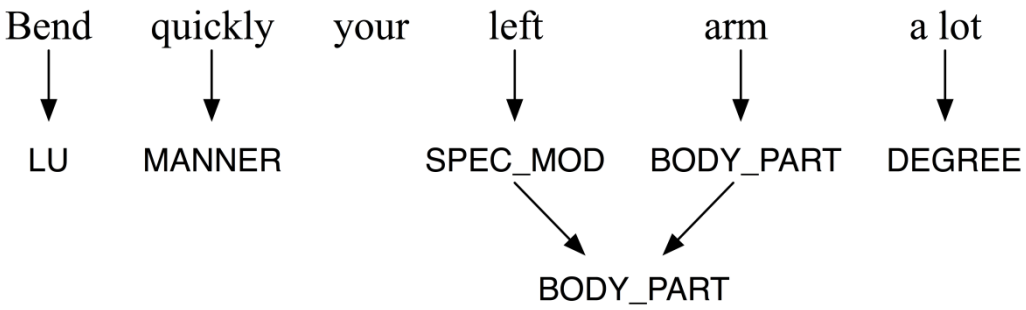
The semantic described here refers to all the actions that the robot can execute and are not related with the environment the robot is acting in. These actions include moving a part of the robot body or moving of a certain quantity of space, with no relation with objects or locations of the surrounding environment.
First, the verbs in the lexicon have been subcategorized according to the frame evoked. For example, body part movement verbs such as

flex, bend, turn, extend, retract...

have been grouped under a new category of terminal symbols called "body_movement_lu" representing the BODY_MOVEMENT frame, as they are potential lexical units for this frame
As for the verbs, the nouns in the lexicon have been divided in new category of terminal symbols according to their semantic function. For example the words related to part of the robot body (i.e. arm, leg, head) have been grouped in terminal category called "body_part" corresponding to the BODY_PART frame element of the frame BODY_MOVEMENT. In the same way, numbers have been specified as "amount" and words like "centimeters" or "degree" as "quantity".

Finally, the subcategory of terminals "specification_modifier" has been defined, including the adjective that specify a property of a noun, like "right" or "left" for the noun "arm".

As described at point A.1.3, the syntactic grammar can be than extended using the semantic tag <tag> and generating a SISR object representing an instantiated semantic frame, as shown below:



FrameNet representation:

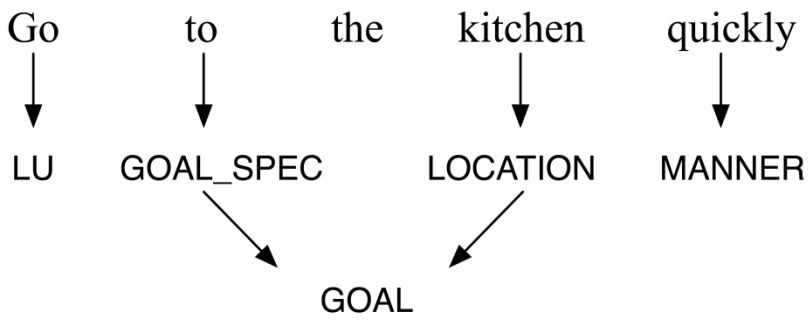
BODY_MOVEMENT:
LU: bend
BODY_PART: left arm
MANNER: quickly
DEGREE: a lot

SISR representation:

```
{
    body_movement: {
        body_movement_lu: bend
        body_part: left arm
        manner: quickly
        degree: a lot
    }
}
```

A.1.3.2 Semantics of the application

The semantic of the application refers to all those entities that are specific of a certain application domain and the action related to the specific environment where the actions take place. As for the *semantics of robot core behavior*, first the verbs have been categorized by evoked frame. For example, verbs like
run, walk, go, move...
evoking the frame SELF_MOTION populated the terminal subcategory of "self_motion_lu".
In the same way, all the nouns in the lexicon representing target entities of the environment have been divided into different semantic category. For example, possible locations represented by words like "kitchen", "bathroom" and "bedroom" have been grouped under the "location" terminal symbols subcategory.
Finally, as described in A.1.3 for the adverbs, the prepositions have been first differentiated between directional and not directional, then the first ones have been specified as done for the adverbs, adding a semantic tag with the directional specifier as "source", "goal" or "direction" that matches with the corresponding frame element.
The result of the overall process is shown in the following example:



FrameNet representation:

SELF_MOTION:
LU: go
GOAL: to the kitchen
MANNER: quickly

SISR representation:

```
{
    self_motion: {
        self_motion_lu: go
        goal: {
            location: kitchen
        }
        manner: quickly
    }
}
```