GENERIC ARCHITECTURE FOR NATURAL LANGUAGE GENERATION IN SPOKEN HUMAN-COMPUTER DIALOGUE

Vladimir POPESCU[*, **], Jean CAELEN[*], Corneliu BURILEANU[**]

[*] Laboratoire d'Informatique de Grenoble, Institut National Polytechnique de Grenoble, France
[**] Faculty of Electronics, Telecommunications and Information Technology, "Politehnica" University of Bucharest, Romania

Corresponding author: Vladimir POPESCU,

Vladimir.Popescu@imag.fr

The human-computer dialogue field is nowadays a rather developed technology and research branch in its own right, but consensus hat not been reached yet with respect to several issues. Out of these, several aspects related to answer generation in spoken natural language are addressed in this paper. First, a modular architecture integrated into a distributed, agent-based dialogue framework and in compliance with existing standard architectures for natural language generation is specified. Its strengths and weaknesses with respect to existing approaches and in the context of human-computer spoken dialogue are emphasized. Second, the interfaces between modules in the architecture are presented in detail; information representation, involving XML grammars, issues are pointed out. Then, several examples of knowledge representation, in XML format, for the information exchanged between the modules in the architecture, are provided. Finally, comparisons with similar work and a set of conclusions are provided.

Key words: spoken language generation; human-computer dialogue; systems architectures, knowledge representation format, XML.

1. INTRODUCTION

Since 2005 our team has started doing research in natural language generation in the framework of spoken human-computer dialogue systems. It is well known that Natural Language Generation is a research field in its own right, but the actual context of spoken dialogue puts forward several specific issues, such as (i) the particularities of the speakers involved in dialogue (in terms of at least social relationship and level of expertise with respect to the topic of the dialogue), (ii) the dynamic evolution of the users' characteristics during dialogue, (iii) the fragmented character of the utterances in dialogue (an utterance from one user can be interrupted by an utterance from the other user taking part in dialogue), (iv) the fact that the language used in dialogue is spoken, hence non-grammatical in the sense that it features ellipses and several other speech-specific characteristics.

The goal of our research resides in the design and development of a generic natural language generation module in a spoken dialogue system; this generation module ought to be as generic as possible, in the sense that:

- domain-specific knowledge should not be involved in language generation,
- language-specific knowledge should not be explicitly encoded in the language generator.

Moreover, the output provided by the language generator should be as expressive as possible, in the sense that, on the one hand, the generated utterances should appear natural and "human-like" to human partners in dialogue and, on the other hand, the output of the generator should allow a speech synthesis module to extract prosodic information conveying emotional states.

In order to achieve a high degree of expressive power and of pertinence (with respect to the context of the dialogue – i.e., the relationship between speakers, their level of familiarity with respect to the topic of the dialogue and the history of the dialogue) for the utterances being generated, resulting in a coherent dialogue, insights from linguistic pragmatics are used, namely, ideas from Speech Act Theory (initially stated by J. L. Austin [1], then augmented by J. R. Searle [2] and formalized in a general semantics framework by Vanderveken [3]) and from discourse representation formalisms – Segmented Discourse Representation Theory (formally stated by N. Asher and A. Lascarides [4], continuing previous work of H.

Kamp in Discourse Representation Theory and of Mann and Thompson in Rhetorical Structure Theory). Thus, the first main insight provided by the Speech Act Theory is that the language can also be used to perform actions, not only to state things; hence one distinguishes, in *performing* an utterance, three types of speech acts: **locutionary** act (corresponding to the actual linguistic statement in the utterance), **illocutionary** act (enclosing the pragmatic – i.e., non-linguistic aim of the utterance) and **perlocutionary** act (specifying the effects of the utterance on its recipient – i.e., the dialogue partner).

In applying previous work in Speech Act Theory (basically, covering only monologues) to dialogue, inter-subjectivity aspects ought to be considered, namely particular features put forward by the two partners in dialogue, through a certain human language and within a certain world (here, shared background knowledge, beliefs, etc.). This framework stems to two poles in dialogue:

- purely linguistic and deictic aspects – represented by language and world – that Searle put forward via the notion of direction of adjustment between words and things;
- inter-subjective aspects – represented by the two speakers as partners in dialogue – that must be considered in dialogue, where speakers adjust their inter-social game.

Given the elements stated above, this paper will present the architecture of the natural language generator for human-computer dialogue, situating it with respect to standard architectures in language generation, emphasizing the knowledge representation issues and design choices for the information exchanged between modules in the architecture. Considerable attention is paid to the representation format for the information exchanged between the modules in our architecture; XML grammars are defined and supported by worked examples. Finally, related work in the field of the architectures for natural language generation is presented, and a set of conclusions and pointers to further work end up the paper.

2. OVERVIEW OF THE ARCHITECTURE

2.1. Processing Levels

One considers that the generation of linguistic utterances in spoken dialogue takes place at five levels that can "produce" one particular aspect of the utterance being output: (i) the logic level, (ii) the pragmatic level, (iii) the linguistic level, (iv) the acoustic level, and (v) the expressive level. Each of these (processing) levels is realized by a sub-module, as pointed out below:

1. the logic level – the logic generator (i.e., the dialogue controller, a module exterior to the natural language generator);
2. the pragmatic level – the pragmatic generator (a sub-module of the natural language generator);
3. the linguistic level – the linguistic generator (a sub-module of the natural language generator);
4. the expressive level – the prosodic generator (a sub-module of the natural language generator);
5. the acoustic level – the speech synthesizer (i.e., a text-to-speech system, exterior to the natural language generator).

In the following lines one will describe the architecture proposed, comparing it to the standard architectures designed by E. Reiter and R. Dale [5].

Thus, the architecture proposed and its parallel to the architecture of Reiter and Dale is presented below and synthesized in figure 1:

1. Logic level – this processing level does not belong to the natural language generation system, but is stated here in order to achieve compatibility to standard architectures in language generation:

- Modules:

- Dialogue controller – Artificial Intelligence-based planification module, based on J. Caelen's work [6];

- Inputs:

- Logic formula, augmented with pragmatic markers (speech acts), corresponding to one user utterance;

- Outputs:

- Logic formula, corresponding to the utterance to be generated by the system;

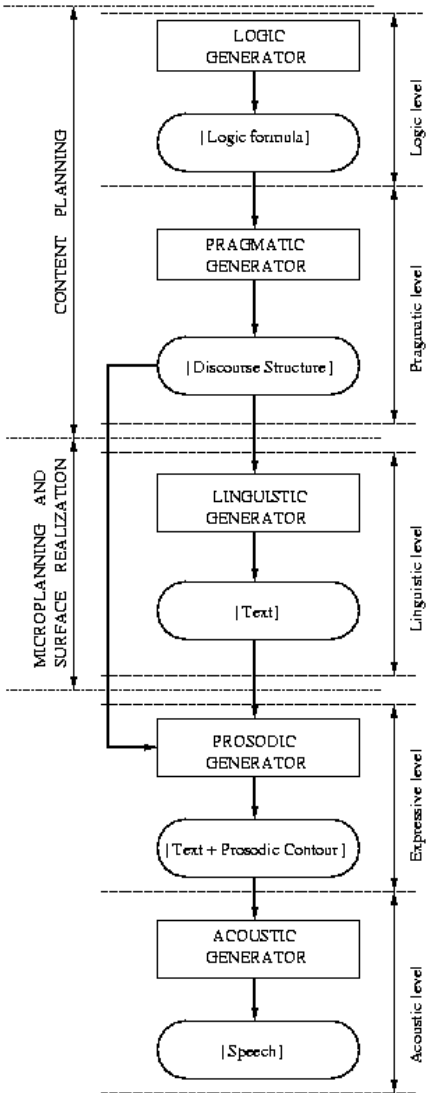

Figure 1. Processing levels, modules and knowledge in the generation system

2. Pragmatic level – this processing level is the first to belong to the natural language generation module and is concerned (in part) in this paper:

- Modules:

- Pragmatic generator – uses discourse representation theories (Segmented Discourse Representation Theory – SDRT [4]) and Speech Acts Theory;

- Inputs:

- Logic formula coming from the dialogue controller, corresponding to the utterance to be generated;
- Logic formula coming from the dialogue controller, corresponding to the utterance previously produced by the user;

- Outputs:

- discourse representation integrating previous dialogue history, for the utterance to be generated;

3. Linguistic level – this processing level uses searching and matching procedures in a corpus-based approach and is not described in this paper:

- Modules:

- Linguistic generator – performs searching and matching in a corpus of annotated spoken dialogues;

- **Inputs**:
  - Logic Formula coming from the dialogue controller, *via* the pragmatic generator;
  - Partial discourse structure coming from the pragmatic generator;
- **Outputs**:
  - Text, corresponding to the utterance to be generated as the response of the dialogue system;

4. **Expressive level** – this level is concerned with the addition of expressiveness to the text previously generated; the expressiveness is considered only with respect to the quality of the speech signal that can be synthesized from the text but, nevertheless, all the processing steps are kept at a textual level:
   - **Modules**:
     - Prosodic generator – this module is responsible with prosodic markup for the text coming from the linguistic level;
   - **Inputs**:
     - Discourse structure coming from the pragmatic level;
     - Text coming from the linguistic level;
     - Degree of illocutionary force for the text;
   - **Outputs**:
     - Prosodic markup, in XML notation;
5. **Acoustic level** – This level performs the actual speech synthesis, in a text-to-speech framework, augmented with features being capable to process prosodic markup coming from the expressive level:
   - **Modules**:
     - Speech synthesizer – this module ought to be able to handle prosodic markup accompanying the text to be synthesized;
   - **Inputs**:
     - Text, augmented with XML markup, stating the variation of prosodic parameters along the utterance to be synthesized;
   - **Outputs**:
     - Synthesized speech.

2.2. Comparison to Standard Architectures in Natural Language Generation

The natural language generation architecture presented in the previous section is here compared to two of the most representative standard architectures in natural language generation, namely, the principled architecture of Reiter and Dale [5] and the more practically-oriented (though, generic as well) RAGS (Reference Architecture for Generation Systems) architecture [7].

Hence, as one can see in Figure 1, the logic and pragmatic levels in our architecture map to the *content planning* module in the architecture of Reiter and Dale, since here the *what-to-say* aspect in generation is handled: the logic level specifies the communicative goal, the speech act to be communicated and the semantic content of the latter, while the pragmatic level reinforces the coherence between the message to be generated and the *history* of the dialogue, providing at the same time the rhetorical structure for the utterance to be generated, still taking into account the history of the dialogue.

Thereafter, one observes a somewhat "reversed" phenomenon: the linguistic level in our architecture maps, at the same time, to the *microplanning* and *surface realization* modules in the architecture of Reiter and Dale, since the linguistic level must handle the generation of the actual textual, linguistic construct for the utterance, out of the non-linguistic input information.

The rest of the levels (and modules) do not map to levels in the architecture of Reiter and Dale, which, in fact, allows for speech generation, mostly via an adaptation of the surface realization module so as to be able to produce speech, in a concept-to-speech framework, hence loosing the flexibility given by the text itself (this flexibility could be important if one wanted to use parts of the generation module in several different applications).

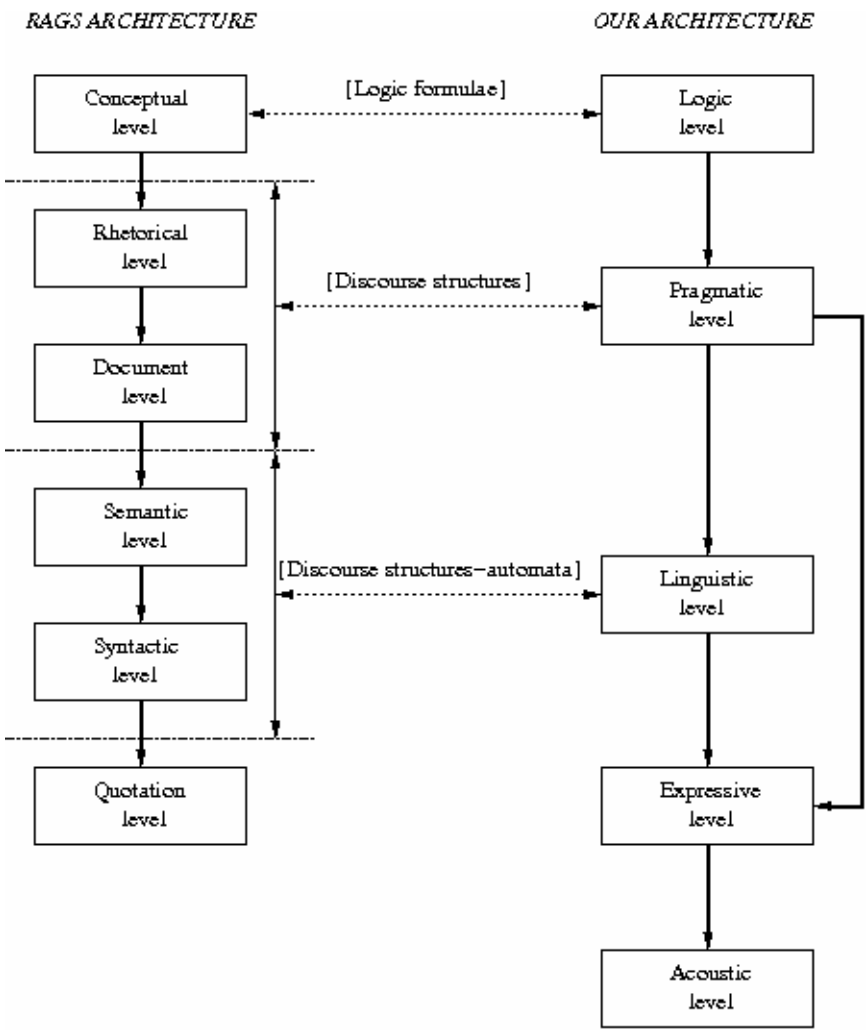

Figure 2. Mappings with the RAGS architecture

Nevertheless, the most important difference between our architecture and that of Reiter and Dale resides in the non-linear processing flow of the former: whereas the architecture of Reiter and Dale supposes an unidirectional processing flow in which one module takes input from and provides output to one and only one module, our architecture allows for the expressive level to be incident to both pragmatic and linguistic levels; moreover, the pragmatic and expressive levels can communicate by bypassing the linguistic level. This type of interaction between modules is not specified in the architecture of Reiter and Dale.

The architecture of Reiter and Dale remains a baseline theoretical framework for natural language generation systems design, but for practical insights that should guide the development of generator in a coherent manner, another architecture seems more appropriate, namely, the RAGS architecture. The latter architecture seems more adapted to practical implementations (since it specifies data representation formats and means to exchange them between several processing levels). Nonetheless, the RAGS architecture still specifies a pipelined architecture, allowing though shortcuts between representation levels, via *non-local* arrows, as pointed out in [7]. Hence the double incidence of the expressive level to other processing levels is compatible, in principle, at a *practical* level, with the RAGS architecture.

The almost one-to-one mappings between RAGS and our architecture are pointed out in figure 2. Thus, the logic level in our architecture maps to the conceptual level in RAGS, but only with respect to language generation aspects, since otherwise the logic generator in our architecture (that is, the dialogue controller) does a lot more than the conceptual level in RAGS [7]. This is why the logic generator in our architecture does not belong to the generation system in itself, constituting instead the interface of the latter system to the dialogue context.

The pragmatic level in our architecture maps at the same time to the rhetorical and to the document levels in RAGS, since the pragmatic generator in our model situates and enforces the coherence of the speech turn to be produced in the context of a dialogue, building at the same time a representation of the discourse, as a discourse structure as shallow as document specifications in a language such as HTML or LaTeX but, of course, without a *linguistic* realization yet.

The linguistic level in our architecture maps to two levels in RAGS too: the semantic and syntactic levels, since in our approach these two processing levels are conflated in a corpus-based semantically and pragmatically-driven approach.

As for the expressive and acoustic levels in our architecture, there is no mapping between these levels and any one level in RAGS, since the aspects related to prosody generation and speech synthesis are not concerned by RAGS, which addresses only the issue of obtaining natural language text as a final result of the generation process.

Therefore, although our architecture has several common points with existing reference architectures in natural language generation, either mostly theoretical as that of Reiter and Dale, or more practical and software-oriented as the RAGS model, remains eclectical in several fine details.

3. INTERFACES BETWEEN MODULES

3.1. Interface between logic and pragmatic levels

The interface between the logic and the pragmatic levels can be viewed under two angles: (i) the static knowledge used by the pragmatic level in situating the information come from the logic level, and (ii) the dynamic information provided by the logic level at each speech turn to be produced by the machine. Thus, the logic level provides dynamic information to the pragmatic level, while the latter uses static knowledge dependent on the current dialogue but not on the current speech turn in dialogue. Hence, there are two types of inputs to the pragmatic generator:

1. **Static** input – represented by prior knowledge in the system; it comprises:
    - social relations between the user and the institution possessing the dialogue system – for instance, client, director, colleague, etc.;
    - level of expertise of the user with respect to the topic of the dialogue;
2. **Dynamic** input – represented by the information come from the logic level during dialogue and comprises logic formulae augmented with speech act and dialogue strategy specifications [6], [19]; hence, this is posterior information for the pragmatic generator, varying along the dialogue.

An important issue in systems architecture design is the specification of the *representation format* of the information exchanged between modules. The pragmatic level in our architecture ought to rely on a consistent and homogeneous representation for the knowledge involved (as inputs); hence, the interface should constitute a *coherent* knowledge set.

As a practical choice for knowledge representation one uses XML markup, following a set of Document Type Definitions (DTD) specifying grammars authorizing the expression of the information exchanged between the modules in our architecture.

Given all the elements above, in specifying the interface between the logic and the pragmatic level the following design choices have been made:

- For the *static* input to the pragmatic generator the following elements are specified in the corresponding DTD:
    - the user "model", stating the social relationship to the dialogue partner and its level of expertise with respect to the topic of the dialogue;
- For the *dynamic* input to the pragmatic generator, the following elements are specified in the corresponding DTD:
    - the speech act for the utterance to be generated;
    - the communicative intention (i.e., a logic formula);

- the dialogue strategy being chosen by the logic generator (a. k. a. the dialogue controller) in order to communicate a certain intention.

All these elements being given, the XML grammar specifying the format of the interface between logic and pragmatic levels is given below:

Elements:

- `USER-TYPE`

this element specifies the type of the user having produced the current utterance; it can be either `MACHINE` or `USER`. Moreover, the type of the user can even not be specified, in order to take into account the situation in which the nature of the user is not (yet) known to the system[1];

- `USER | MACHINE`

each element `USER` or `MACHINE` is `EMPTY`, hence not containing other elements;

- `SPEECH-ACT`

this element specifies the speech act and can be either `EMPTY` or represented by an element of the type predicate;

- `PREDICATE`

this element can either be explicitly given (`#PCDATA`), or be obtained via a *reference* to a logic formula, specified in XML Common Logic (XCL) language (`%LOGIC-FORMULA`). Moreover, this element can be `EMPTY`, thus taking into account the absence of the speech act to communicate;

- `STRATEGY`

this element specifies the strategy chosen at the logic level by the dialogue controller in order to communicate a certain speech act. This element does not contain other sub-elements, having instead several attributes that one will see further;

- `GOAL`

this element specifies the goal put by the dialogue controller when communicating the speech act; this goal has to be a part of the output provided by the logic level to the pragmatic level, so that the latter can used this goal in the computation of the illocutionary force, for instance. This goal can be either explicitly stated (#PCDATA), or referenced to a predicate (`%LOGIC-FORMULA`); it can even be `EMPTY`, thus marking the absence of the goal.

Attributes:

- `MACHINE | USER::relation`

this attribute specifies the relationship to the dialogue partner; the value for this attribute must be stated (`#REQUIRED`); the default value is `equal`;

- `MACHINE | USER::level`

this attribute specifies the level of expertise for the emitter of the current speech act; the value for this attribute must be specified, the default being `expert`;

- `MACHINE | USER::identifier`

this attribute is used for the internal knowledge management and its value need not be specified, but is initialized by the system (`#IMPLIED`);

- `SPEECH-ACT::source`

this attribute specifies the emitter of the current speech act; its value must be stated;

- `SPEECH-ACT::destination`

similar to the preceding attribute, the destination speaker for the current speech act is specified; the value for this attribute must be stated;

- `SPEECH-ACT::type`

this attribute specifies one of the six types of speech acts, as formalized by J. Caelen [6]; its value must be stated;

[1] This way, the specification of the interface between logic and pragmatic levels allows for dialogue between a machine and either a human user or another machine.

- `STRATEGY::type`

this attribute specifies one of the six strategies used in conveying communicative intents to speakers, in dialogue and can take one value according to the formalization provided by J. Caelen [6]; the value for this attribute must be stated;

- `STRATEGY::governor`

this attribute specifies the speaker having the initiative in the strategy used to convey a certain speech act; this attribute must be initialized;

- `STRATEGY::identifier`

this attribute is implicitly initialized by the system and its value its value need not be provided by the user; its purpose is to identify a certain strategy *token* in the system;

- `GOAL::initiator`

this attribute specifies the speaker having established the dialogue goal being currently resolved; its value must be provided by the user;

- `GOAL::scope`

this attribute specifies the scope of the goal (i.e., wether the goal is a main goal or a sub-goal) and its value need not be provided by the user, since it results from the internal operation of the system;

- `GOAL::state`

this attribute specifies the state of the goal (in the framework provided by work of J. Caelen [6]), i.e., “stated”, “attained”, “satisfied”, “pending”, “repaired”, “abandoned”, and its value must be provided;

- `GOAL::identifier`

this attribute is implicitly initialized by the system and its value its value need not be provided by the user; its purpose is to identify a certain goal *token* in the system;

Entities:

The DTD specifying the interface between logic and pragmatic levels gains access to the XML representation of the logic formulae through a reference to another DTD specifying the logic formulae[2]; hence, the following entity has been defined:

- `LOGIC-FORMULA`

this entity represents the reference to the DTD specifying the grammar for the logic formulae received from the dialogue controller, expressed in first-order predicate logic.

3.2. Interface between pragmatic and linguistic levels

The pragmatic generation level computes a discourse structure for each dialogue being undergone. However, only the speech act currently due to be put in linguistic form has to be specified to the linguistic generation level. The speech act to “generate” (i.e., to put in textual form) has to be in a context known to the linguistic generator, since the latter operates in a corpus-based manner, performing search and matching on such (discourse) contexts, but its operation is not described in this paper. In order to provide the necessary information to the linguistic generation level, the pragmatic level provides a SDRS discourse structure containing rhetorical relations between utterances, represented in turn as labels accompanied by logic formulae. Thus, the complete output provided by the pragmatic level as input to the linguistic level is also represented in XML format, following a specific DTD that will be presented below.

The design of the XML grammar for expressing is guided by the fact that the information needed by the linguistic generation level is the following:

1. At the level of one utterance:

[2] In fact, a slightly modified version of the “XML Common Logic” (XCL) language is used [18].

- the speaker having produced the utterance (the human user), or due to produce it (the machine);
- the dialogue strategy used to produce the current utterance;
- the speech act realized by the current utterance;
- the degree of strength for the illocutionary force characterizing the current utterance;

2. At the level of sets of utterances:
   - utterance labels, one label for each utterance;
   - rhetorical relations between pairs of utterances, within the framework of the SDRT (the set of rhetorical relations used will not be detailed in this paper);
3. At extra-discourse level, the user model, come from the *static input* to the pragmatic generator:
   - the level of expertise of the user with respect to the topic of the dialogue;
   - the social relationship between the user and the institution owning the dialogue system.

All these elements given, the XML grammar for the interface between pragmatic and linguistic levels is specified in the DTD presented below:

Elements:

- `DIALOG`
  this element specifies an instance of dialogue;
- `SPEAKER`
  this element specifies the speaker having produced or due to produce an utterance;
- `ACT`
  this element specifies a speech act for an utterance already produced or due to be produced;
- `STRATEGY`
  this element specifies the dialogue strategy used in realizing an utterance;
- `RRELATION`
  this element specifies a rhetorical relation connecting two utterances in a dialogue;
- `SDRS`
  this element specifies a discourse representation structure for a dialogue or a fragment of a dialogue;
- `STATE`
  this element specifies a building block a discourse representation structure; it can be either an utterance or a discourse sub-structure [4]; the terminology is inspired from a customized automata-based formalism designed by our team in order to perform stochastic handling of the discourse representation structures [19]; this formalism is not described in this paper;
- `UTTERANCE`
  this element specifies a particular utterance in a discourse structure contained in a dialogue;
- `FORMULA`
  this element specifies the logic formula for a particular utterance in a dialogue;

Attributes:

- `SPEAKER::type`
  this attribute specifies the type of the speaker, taking one of the values user or machine; its value must be specified;
- `SPEAKER::name`
  this attribute specifies a conventional name for a particular speaker; its value need not be specified;
- `SPEAKER::level`
  this attribute specifies the level of expertise for a particular speaker, with respect to the topic of the current dialogue; its value must be provided;
- `SPEAKER::relation`

this attribute specifies the social relationship between a particular speaker and its dialogue partner; its value must be provided;

- ACT::type

this attribute specifies the type of the speech act realizing a certain utterance; its value must be provided, and can be one of the following: F, FF, FS, FFS, FP, FD, following the taxonomy provided by J. Caelen [6];

- ACT::name

this attribute specifies a name for a speech act type; it can be a gloss on the speech act type; its value need not be provided;

- DIALOG::time

this attribute specifies the time of the dialogue, in order to allow for chronological sorting of the dialogues; its value is a matter of convention (#PCDATA), but must be provided;

- STRATEGY::type

this attribute specifies the type of the dialogue strategy used in order to produce an utterance; its value must be provided and can be one of the following: D (directive strategy), R (reactive strategy), N (negociative strategy), C (constructive strategy), K1 (cooperative strategy – expert user case), K2 (cooperative strategy – novice user case);

- STRATEGY::name

this attribute specifies a name for a strategy type; it can be a gloss on the strategy type and its value need not be provided;

- RRELATION::type

this attribute specifies the type of the rhetorical relation connecting two utterances; its value must be provided and can be one of the following 17: QELAB ("Question Elaboration"), IQAP ("Indirect Question-Answer Pair"), PCORR ("Plan Correction"), PELAB ("Plan Elaboration"), BACKGQ ("Background$_q$"), ELABQ ("Elaboration$_q$"), NARRQ ("Narration$_q$"), QAP ("Question-Answer Pair"), ACK ("Acknowledgement"), NEI ("Not Enough Information"), ALT ("Alternation"), BACKGR ("Background"), CONSEQ ("Consequence"), ELAB ("Elaboration"), NARR ("Narration"), CONTR ("Contrast"), and PARALL ("Parallel");

- RRELATION::name

this attribute specifies a name for a rhetorical relation type; it can be a gloss on the rhetorical relation type and its value need not be provided;

- SDRS::label

this attribute specifies a label for a discourse representation structure; its value is a matter of convention (#PCDATA) and must be provided;

- SDRS::type

this attribute specifies the type of a discourse representation structure; its value must be provided and can be either sdrs or enounce;

- SDRS::nstates

this attribute specifies the number of utterances and discourse sub-structures within a discourse structure; a labeled discourse sub-structure is a considered a state in a current discourse structure; the value of this attribute must be provided;

- SDRS::nrels

this attribute specifies the number of rhetorical relations connecting pairs of utterances and discourse sub-structures in a discourse structure; its value must be provided;

- STATE::type

this attribute specifies the type of a state in a current discourse structure; its value must be provided and can be either SDRS (for a set of rhetorical relations in a discourse sub-structure) or UTT (for a single utterance);

- STATE::label

this attribute specifies the label of a state in a current discourse structure; its value must be provided;

- `UTTERANCE::speaker`

this attribute specifies the speaker having produced or due to produce the current utterance; its value must be provided and must be the name of the speaker (as specified by the attribute `SPEAKER::name`);

- `UTTERANCE::speech-act`

this attribute specifies the speech act used to communicate the current utterance; its value must be provided;

- `UTTERANCE::strategy`

this attribute specifies the dialogue strategy used in order to produce the current utterance; its value must be provided;

- `UTTERANCE::illoc-force`

this attribute specifies the degree of strength for the illocutionary force characterizing the current utterance;

Entities:

- `XCL-formula`

this entity represents a reference to a logic formula expressed in XCL language;

- `SPEAKERS`

this entity gives the set of speakers as partners in a dialogue;

- `ACTS`

this entity gives the set of known speech act types;

- `STRATEGIES`

this entity gives the set of known dialogue strategy types;

- `RRELATIONS`

this entity gives the set of known rhetorical relations types.

3.3. Interface between pragmatic, linguistic and expressive levels

The inputs to the expressive level are made of information provided by the pragmatic generation level to the linguistic level, accompanied by the text provided by the linguistic level as its output. Therefore, the principles and the XML specifications presented in the preceding two sub-sections apply also for representing the information exchanged between the pragmatic and linguistic levels, and the expressive level.

3.4. Interface between expressive and acoustic levels

As stated in section 2, in our architecture the acoustic generation level is a speech synthesizer able to accept and process prosodic markup provided, in textual form, by the expressive level. Hence, the inputs to the acoustic level are, on the one hand, the text come from the linguistic level and, on the other hand, prosodic markup come from the expressive level.

Assuming that the prosodic markup provided by the expressive level concerns only the contour of the fundamental frequency curve, this latter one, denoted by Ψ_0, is expressed in terms of a set of seven *elementary* contours, denoted by "++", "+", "+=", "=", "-+", "-", "--". Thus, denoting by "↑" the concatenation operation, the contour of the fundamental frequency for a particular speech signal is expressed as:

$$\left\langle \Psi_0 \right\rangle = \uparrow_{i=1}^{N} \left\{ x_i^{(0)} \% \times \left(++;+;+=;=;-=;-;-- \right) \times v_i^{(0)} \right\} \tag{1}$$

where $v_i^{(0)}$ is a vector of module 1 in the real orthogonal base of seventh order, and $x_i^{(0)}$ are positive real numbers such that:

$$\sum_{i=1}^{N} x_i^{(0)} = 100 \tag{2}$$

and *N* is a user-defined parameter determined in an empiric manner; this number specifies the number of points in which the fundamental frequency contour is expressed. Its value is chosen during experiments, according to the application, the speech corpora being used and the results expected.

As for the meaning of the elementary contours, this is specified as follows: (i) "++" denotes a signal of slope 2 and unit duration, (ii) "+" denotes a signal of slope 1 and unit duration, (iii) "+=" denotes a signal of slope 0.5 and unit duration, (iv) "=" denotes a signal of slope 0 and unit duration, (v) "-=" denotes a signal of slope -0.5 and unit duration, (vi) "-" denotes a signal of slope -1 and unit duration, (vii) "--" denotes a signal of slope -2 and unit duration. The hypothesis behind this representation scheme is that the fundamental frequency has a sufficiently smooth variation to be represented via these seven elementary contours, weighted by percents of the total acoustic signal length; these percents give the duration for each elementary contour.

This manner of representing fundamental frequency contours are inspired from and related to formalisms such as the one designed by J. Pierrehumbert, or ToBI [8], [10]. Nevertheless, these representation conventions need to be specified in an XML format in order to constitute an interface between expressive and acoustic generation levels.

The fundamental frequency contour is mapped to an utterance for which the text is given by the linguistic generation level; in order to perform this mapping in a transparent manner, an XML grammar, presented below, has been defined.

Elements:

- `UTTERANCE`

this element specifies the current utterance; it contains the text and the corresponding prosodic markup;

- `TEXT`

this element specifies the raw text (#PCDATA) for the current utterance;

- `CONTOUR`

this element specifies the fundamental frequency contour as a sequence of weighted elementary contours, indexed on a time scale;

- `NNE`, `NE`, `ENE`, `E`, `ESE`, `SE`, `SSE`

these elements specify respectively the elementary contours "++","+", "+=", "=","-=", "-", "--";

Attributes:

- `UTTERANCE::length`

this attribute specifies the length (in seconds) of the acoustic realization for the current utterance; its value must be provided;

- `UTTERANCE::id`

this attribute identifies the utterance in the system; its value need not be provided;

- `CONTOUR::npoints`

this attribute specifies the number of points in which the fundamental frequency contour is expressed; its value must be provided;

- `CONTOUR::id`

this attribute identifies the fundamental frequency contour for the current utterance; its value need not be provided but must be identical to the value of the attribute `UTTERANCE::id`;

- `{NNE | NE | ENE | E | ESE | SE | SSE}::percent`

this attribute specifies the vector of percents (weights) for each elementary contour in the current utterance; its values must be provided;

- `{ NNE | NE | ENE | E | ESE | SE | SSE }::index`

this attribute specifies the indexes of the points (given by the attribute `CONTOUR::npoints`) that the elementary contour referred occupies (in order) in the current utterance.

3.5. Examples

In this section we will present an extensive example of information representation using the interfaces previously defined. Thus, we will assume that a fragment of transcribed spoken dialogue is available, and that a certain utterance (the last one) is due to be produced by the machine. Hence, the discourse structure, in the framework of the SDRT, will be represented, as well as the logic formula for the utterance to be generated, accompanied by the dialogue strategy and the speech act for this latter utterance.

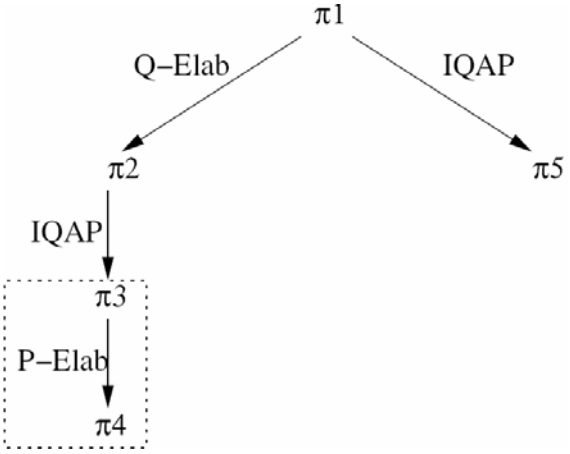

Figure 3. Discourse structure for a dialogue

Therefore, we assume that the following fragment of dialogue is available (by "*U*" and "*M*" we denote the human user and the machine, respectively), and π_1, …, π_5 are the labels of the utterances in the dialogue shown below:

M: π_1 : And you want whose book on "X"?

U: π_2 : Well, what's available?

M: π_3 : We have authors "Y" and "Z".

M: π_4 : We had also authors "T" and "W" yesterday…

U: π_5 : So, I would choose author "Z"!

The discourse structure, in the SDRT formalism, is shown in figure 3, will be represented in XML format, in order to be passed from the pragmatic generation level to the linguistic level in our architecture. Moreover, we suppose that the machine (*M*) is due to generate an answer of confirmation to the last statement coming from the user (*U*), that will correspond to an utterance labeled π_6.

Hence, the logic generation level will produce a communicative intent expressing the confirmation of the fact that the user can indeed choose (for buying in the case of a book store, or borrowing in the case of a library) the book of author "Z"; a logic formula, in first-order predicate logic, would be as expressed below:

$$\exists \text{book}(X) \wedge \text{author}(X, y) \wedge \text{equals}(y, \text{"}Z\text{"}) \wedge \text{choose}(user, X, ok) \wedge \text{equals}(user, U) \wedge \text{equals}(ok, 1) \quad (3)$$

where book is a unary predicate, author and equals are binary predicates, and choose is a ternary predicate.

The pragmatic generation level takes as input the discourse structure as shown in figure 3, the logic formula as shown in equation (3), assigns the label π_6 to it and adds it to the discourse structure in figure 3, linking π_6 to π_5 via the rhetorical relation ACK(π_5, π_6). In order to do that, it uses the information on the user profiles, namely *U*(client, expert), and *M*(institution, expert), the strategy used in communicating π_6, namely cooperative-expert case (K1), and the speech act used in conveying π_6 to the user, namely FS (DO-

KNOW). Hence, the input to the pragmatic generator can be expressed, in XML, for the utterance to be generated (π_6), as below:

```
<USER-TYPE>
       <MACHINE relation=”server” level=”expert” identifier=”ID7”/>
</USER-TYPE>
<SPEECH-ACT source=”machine” destination=”user” type=”FS” identifier=”I9”>
       <PREDICATE>
              <quant name=”exists” variable=”X”>
                     <type>book</type>
                     <conn name=”and”>
                            <pred name=”author”>
                                   <term name=”X”/>
                                   <term name=”y”/>
                            </pred>
                            <pred name=”equals”>
                                   <term name=”y”/>
                                   <term name=”Z”/>
                            </pred>
                            <pred name=”choose”>
                                   <term name=”user”/>
                                   <term name=”X”/>
                            </pred>
                            <pred name=”equals”>
                                   <term name=”user”/>
                                   <term name=”U”/>
                            </pred>
                            <pred name=”equals”>
                                   <term name=”ok”/>
                                   <term name=”1”/>
                            </pred>
                     </conn>
              </quant>
       </PREDICATE>
</SPEECH-ACT>
<STRATEGY type=”cooperative-1” governor=”all” identifier=”ID09”/>
<GOAL initiator=”user” scope=”goal” state=”attained” identifier=”ID09”>
       <pred name=”choose”>
              <term name=”book”/>
       </pred>
</GOAL>
```

The linguistic generation level takes as input the discourse structure updated by the pragmatic level, as stated above, along with the logic formulae for the utterances in the discourse structure and outputs a textual representation for π_6 , such as:

M: π_6 : OK, you can take the book by the author “Z”!

Hence, the input to the linguistic generator contains, besides the previous statement for the communicative intents “behind” the utterances in the discourse structure, the XML representation for the discourse structure updated from that in figure 3, so that in integrates the utterance π_6 as below:

```
<SDRS label=”pi0” nstates=”5” nrels=”4”>
       <STATE label=”pi1” type=”UTT”>
              <UTTERANCE speaker=”MACHINE” strategy=”K1” speech-act=”FFS”
                         illoc-force=”72”>
                     And you want whose book on “X”?
              </UTTERANCE>
              <FORMULA>
                     <!-- XCL code -->
              </FORMULA>
       </STATE>
       <!-- Other states of the type “UTT” -->
```

```
      <STATE label="pi3prime" type="SDRS">
            <SDRS label="pi3prime" nstates="2" nrels="1">
                  <STATE label="pi3" type="UTT">
                        <!-- Definition for the utterance "pi3" -->
                  </STATE>
                  <!-- Definition for state "pi4" -->
            </SDRS>
      </STATE>
      <STATE label="pi6" type="UTT">
            <UTTERANCE speaker="MACHINE" strategy="K1" speech-act="FS"
                  illoc-force="31">
                  NULL
            </UTTERANCE>
            <FORMULA>
                  <!-- The formula specified in the input to the pragmatic generation
                        level -->
            </FORMULA>
      </STATE>
      <RELATION label="rho1" type="QELAB" left="pi1" right="pi2"/>
      <RELATION label="rho2" type="IQAP" left="pi2" right="pi3prime"/>
      <!-- Other rhetorical relations, as shown on figure 3 -->
      <RELATION label="rho5" type="ACK" left="pi5" right="pi6"/>
</SDRS>
```

The expressive generation level takes the text produced by the linguistic level, along with the discourse structure provided by the pragmatic level and outputs prosodic markup associated to the text, expressed in XML format as below:

```
<UTTERANCE length="4" id="ID09">
      <TEXT>
            OK, you can take the book by the author "Z"!
      </TEXT>
      <CONTOUR npoints="512" id="ID09">
            <NNE percent="4.75;5.37;7.33;9.00;0.73" index="3;5;7;152;365"/>
            <!-- The other the fundamental contours with their weights and indexes -->
      </CONTOUR>
</UTTERANCE>
```

This prosodic markup is taken by the acoustic generation level as input; the acoustic level outputs in its turn the speech signal for the utterance π_6.

4. RELATED WORK

The idea of designing an architecture for natural language generation has already been mitigated in several instances, out of which the most interesting seem to be the standard architecture proposed by Reiter and Dale and the RAGS architecture. Nevertheless, none of these models was targeted towards integration in spoken human-computer dialogue systems, researchers having approached the problem of natural language generation for dialogue in several fashions, varying from purely rule-based approaches [16], passing through grammar-based approaches [17], ad-hoc combined approaches [15], or purely probabilistic, principled, approaches [13], [14].

As a general remark concerning these approaches, they usually lack genericity with respect to application domain or language being used; that is precisely what our architecture tries to mitigate.

5. CONCLUSIONS AND FURTHER WORK

Our paper has described a generic (independent of the application domain and language used) architecture for spoken language generation in human-computer dialogue, where the language generation process is viewed as a chain of several stages or generation levels, namely, (i) the logic level, (ii) the

pragmatic level, (iii) the linguistic level, (iv) the expressive level, and (v) the acoustic level. Each of these levels is characterized by three features:

- a set of inputs, specified in a consistent manner, following suited XML grammars;
- a module instantiating the processing of the inputs, in order to obtain results to be output;
- a set of outputs, specified in a consistent manner as well.

While the set of modules is compatible, in principle, with the standard architectures of Reiter and Dale and with the RAGS reference model, the inputs and outputs to and of these modules are defined in a precise manner, which is, though, dependent on the underlying theoretical assumptions driving the inner working of these modules. However, the way that the interfaces are defined is not related to the way in which this information is actually processed within the modules, a certain degree of separation between the modules (via the interfaces between them) being thus achieved.

The way in which our architecture is defined allows for a rather straightforward software design, in an object-oriented manner, using for instance UML or other similar modelling languages[3].

A step further in refining the architecture presented in this paper would be to render it compatible with the RAGS model, by aligning the XML interface specifications provided in our architecture to those provided in RAGS. More specifically, we need to attentively customize the (rather) free notation provided in RAGS to ours, which relies on specific theoretical assumptions driving the processing steps accomplished at each generation level.

The compliance of our architecture to the RAGS specification would open the way to consistent evaluations of our architecture with respect to other natural language generation architectures already complying with the RAGS model (for example, the IDAS system [10]), which is very important for performance comparisons driving decisions on which architectures to adopt when building a particular natural language generation system.

REFERENCES

1. AUSTIN, J.L., *How to Do Things with Words*, Oxford University Press, 1962.
2. SEARLE, J. R., *Speech Acts*, Cambridge University Press, 1969.
3. VANDERVEKEN, D., *Meaning and Speech Acts*, Cambridge University Press, 1990-1991.
4. ASHER, N., LASCARIDES, A., *Logics of Conversation*, Cambridge University Press, 2003.
5. REITER, E., DALE, R., *Building Natural Language Generation Systems*, Cambridge University Press, 2000.
6. CAELEN, J., "Strategies of Dialogue", *Proc. Of The Second Conference on Speech Technology and Human-Computer Dialogue SpeD 2003*, Publishing House of the Romanian Academy, Bucharest, 2003.
7. MELLISH, C., SCOTT, D., CAHILL, L., PAIVA, D., EVANS, R., REAPE, M., "A Reference Architecture for Natural Language Generation Systems", *Natural Language Engineering*, 12 (1): 1-34, Cambridge University Press, 2006.
8. BOD, R. HAY, J, JANNEDY, S. (Eds.), *Probabilistic Linguistics*, MIT Press, 2003.
9. MANNING, C., SCHUTZE, H., *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.
10. JURAFSKY, D., MARTIN, J. H., *Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, Prentice Hall, 2000.
11. RUSSELL, S, NORVIG, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2003.
12. CHAMBERS, N., "Real-Time Stochastic Language Generation for Dialogue Systems", *Proceedings of the 10th International Workshop on Natural Language Generation*, Aberdeen, 2005.
13. OH, A., RUDNICKY, A., "Stochastic Natural Language Generation for Spoken Dialog Systems", *Computer Speech and Language*, vol. 16, 2002.
14. RATNAPARKHI, A., "Trainable Approaches to Surface Natural Language Generation and their Application to Conversational Dialog Systems", *Computer Speech and Language*, vol. 16, 2002.
15. STENT, A., "A Conversational Acts Model for Generating Spoken Dialogue Contributions", *Computer Speech and Language*, vol. 16, 2002.
16. STONE, M., DORAN, C., WEBBER, B., PALMER, M., "Microplanning with Communicative Intentions: the SPUD System", *Computational Intelligence*, vol. 19, 2003.
17. THEUNE, M., *From Data to Speech: Language Generation in Context*, Ph D Thesis, University of Eindhoven, 2000.
18. ALTHEIM, M., *XML Common Logic (XCL) 1.0. – a Concrete Syntax for Common Logic in XML*, http://purl.org/xcl/1.0/, 2003.
19. POPESCU, V., "Architecture pour la génération du langage naturel en dialogue homme-machine", *Ecole thématique du CNRS "Dialogue et Interaction"* , Autrans, 2006.

[3] In fact, our architecture has already been designed in UML and, in part, implemented in C++ programming language.