Reusing MT Components in Natural Language Generation for Dialogue Systems[*]

Gabriel Amores Guillermo Pérez Pilar Manchón
G. Investigación Julietta
Universidad de Sevilla
jgabriel,gperez,pmanchon@us.es

Resumen: Este artículo describe investigación en curso sobre generación de lenguaje natural para sistemas de diálogo. Una serie de plantillas se encargan de la fase de planificación clausal, mientras que unos módulos de transferencia y generación desarrollados previamente para un sistema de TA llevan a cabo los procesos de lexicalización y realización morfosintáctica. Este enfoque favorece la generación multilingüe, al separar claramente la información dependiente del lenguaje de aquella que no lo es.
Palabras clave: Generación de Lenguaje Natural, Sistemas de gestión de Diálogo.

Abstract: This paper describes ongoing research on NLG for dialogue systems. Sentence planning is performed by selecting the appropriate template, while previously developed transfer and generation components of a transfer–based MT architecture perform the lexicalization and linguistic realization processes in the generation process. This approach allows for multilingual generation since there is a clear division between language dependent and language independent information.
Keywords: Natural Language Generation, Spoken Dialogue Systems.

1 *Introduction*

This paper describes ongoing research on Natural Language Generation in the Delfos spoken dialogue system (Amores and Quesada, 2001).

Our previous version of Delfos in the European DHomme (Dhomme, 2001) and Siridus (Siridus, 2002) projects synthesized canned text and/or pre–recorded audio files. Those projects focused on how the Information State Update Approach (Traum and Larsson, 2001) to dialogue management in which Delfos is inspired exhibited a much higher degree of dialogue flexibility and naturalness. While Natural Language Generation was not a priority in those projects, the Talk project (Talk, 2004) currently under way includes a work package especially devoted to how information should be presented to the user in a multimodal setting.

According to the ISU approach, the dialogue information state is enriched with grounded and expected information as the dialogue progresses. The hypothesis underlying our work is that all the information necessary to generate natural language is available to the system when it is requested to produce a spoken message. In particular, this paper describes how a natural language expression is obtained from a **DTAC** information state. DTAC stands for the **D**(ialogue) move, **T**ype, **A**rguments and **C**ontent features in our dialogue information state.

1.1 Natural Language Generation

Natural language generation (NLG) is the process of automatically creating natural language from a non–linguistic information representation. This process has traditionally been conceived as involving variations of the following phases (Reiter, 1994) (Reiter and Dale, 2000), (DeSmedt, Horacek, and Zock, 1996):

Content Determination maps the initial input of the generation component onto a semantic form.

Sentence Planning maps conceptual structures onto linguistic ones.

Linguistic Realization takes as input an abstract specification of information to be communicated by syntax and function words, and produces as output a

[*] This research has been funded by the Spanish Ministry of Education under Grant TIC2002-00526, and the European FP6 IST Talk Project (507802).

surface form that communicates this information, including morphological realization.

Formatting contains mechanisms for formatting and/or adding some type of artificial language tag (SSML, hypertext, etc.)

In general, it can be said that the tasks of phase 1 are language–independent but domain specific, whereas linguistic realization (phase 3) is language–specific. The tasks associated with sentence planning usually require both domain and language–specific knowledge. However, as we will see in our system, our sentence planner generates a language–independent structure, which is then transferred to a language specific intermediate representation.

1.2 Natural Language Generation in Dialogue Systems

In a dialogue system, the input for the NLG component generally consists of a message specification produced by the dialogue manager, which describes the content to be expressed in the system's next dialogue turn. This means that it is the dialogue manager which decides what to say and when to say it, thus performing the generation tasks of *content determination* and *discourse planning.* In many practical dialogue systems there appears to be no direct need for a full–fledged language generation component, since relatively simple techniques are sufficient to produce adequate language output. The two current approaches to linguistic realization in dialogue systems are template–based and rule–based (linguistic) NLG. Several grammar–based generation systems and platforms have been developed such as KPML (Bateman, 1997), FUF/Surge (Elhadad and Robin, 1997) and RealPro (Lavoie and Rambow, 1997). The use of these platforms for dialogue systems has its disadvantages for practical reasons since a general purpose generator must be simplified for the domain at task.

In the current state of the art, template–based realization seems to be more attractive from a practical point of view, and it is the preferred type of generation in spoken dialogue systems. A number of them have been developed in recent years, such as TG/2 (Buseman and Horacek, 1998), YAG (McRoy, Channarukul, and Ali, 2003), GENESIS-II (Baptist and Seneff, 2000), XTRAGEN (Stenzhorn, 2002), D2S ((Theune et al., 2001), (van Deemter and Odijk, 1997)), and EXEMPLARS (White and Caldwell, 1998). Nevertheless, most template–based systems are designed for a specific language and a specific domain. When a template–based system is to be used in a different application or another language, most of the templates will have to be replaced.

Finally, some other systems have opted for a statistical approach (Oh and Rudnicky, 2000) (Rambow, Bangalore, and Walker, 2001) (Zhong, 2004) applying machine learning techniques which require far less time for tuning than both template–based and grammar–based approaches.

This paper focuses on how the process of sentence planning can be made more language independent by reusing previously developed transfer and generation components in a MT architecture.

The paper is organized as follows. Section 2 briefly describes the domain and scenario for which the generation system is being developed. Next, section 3 outlines how a series of templates[1] are specified and called from a dialogue rule in the dialogue manager. Section 4 describes how a transfer and generation module in a MT system may be reused to accomplish the linguistic realization phase in the overall generation process. Finally, section 5 discusses the advantages and limitations of the current approach, while section 6 outlines future work.

2 The Home Domain

In the home domain under discussion, a user has multimodal control (i.e. by speech and/or clicks) over a set of devices in the home (lights, telephone, etc.). Depending on the context, the dialogue situation, and the channels available, the system will decide the type and amount of information to be conveyed to the user. In this paper, we will just consider a speech–in speech–out scenario.

As a working example, consider the following situation. Upon the user's request *How many lights are switched on in the kitchen?* the system obtains the following information state:[2]

[1]Our notion of template here should not be confused with template–based approaches to realization

[2]Features have been truncated to facilitate readability.

$$\left[\begin{array}{ll}
\text{DMOVE} & \text{specComm} \\
\text{TYPE} & \text{ReqQuant} \\
\text{ARGS} & \left[\text{DevSpec}\right] \\
\text{DevSpec} & \left[\begin{array}{ll}
\text{DMOVE} & \text{specPar} \\
\text{TYPE} & \text{DevSpec} \\
\text{ARGS} & \left[\text{Loc,DevType,State}\right] \\
\text{Loc} & \left[\begin{array}{ll} \text{DMOVE} & \text{SpecPar} \\ \text{TYPE} & \text{Loc} \\ \text{CONT} & \text{KITCHEN} \end{array}\right] \\
\text{DevType} & \left[\begin{array}{ll} \text{DMOVE} & \text{SpecPar} \\ \text{TYPE} & \text{DevType} \\ \text{CONT} & \text{LIGHT} \end{array}\right] \\
\text{State} & \left[\begin{array}{ll} \text{DMOVE} & \text{SpecPar} \\ \text{TYPE} & \text{State} \\ \text{CONT} & \text{ON} \end{array}\right]
\end{array}\right]
\end{array}\right]$$

After reference resolution, the dialogue manager returns an augmented DTAC structure with the relevant information. In this example, two lights have been found:

$$\left[\begin{array}{ll}
\text{DMOVE} & \text{specCom} \\
\text{TYPE} & \text{ReqQuant} \\
\text{ARGS} & \left[\text{DevSpec}\right] \\
\text{DevSpec} & \left[\begin{array}{ll}
\text{DMOVE} & \text{specPar} \\
\text{TYPE} & \text{DevSpec} \\
\text{ARGS} & \left[\text{Loc,DevType,State}\right] \\
\text{Loc} & \left[\begin{array}{ll} \text{DMOVE} & \text{SpecPar} \\ \text{TYPE} & \text{Loc} \\ \text{CONT} & \text{KIT} \end{array}\right] \\
\text{DevType} & \left[\begin{array}{ll} \text{DMOVE} & \text{SpecPar} \\ \text{TYPE} & \text{DevType} \\ \text{CONT} & \text{LIGHT} \end{array}\right] \\
\text{State} & \left[\begin{array}{ll} \text{DMOVE} & \text{SpecPar} \\ \text{TYPE} & \text{State} \\ \text{CONT} & \text{ON} \end{array}\right] \\
\text{RefRes} & \left[\begin{array}{ll} \text{Quant} & 2 \\ \text{RR1} & \left[\begin{array}{ll} \text{devId} & \text{A1} \\ \text{Loc} & \text{KIT} \end{array}\right] \\ \text{RR2} & \left[\begin{array}{ll} \text{devId} & \text{A2} \\ \text{Loc} & \text{KIT} \end{array}\right] \end{array}\right]
\end{array}\right]
\end{array}\right]$$

3 *Sentence Planning*

Taking the example above as an illustration, we can notice that the information state contains all the information necessary to produce the desired output message. Sentence planning takes the DTAC structure above and returns a feature structure with linguistic features instead, by means of predefined generation templates. Given the information state above, a function is called with specifies the current information state (@IS), and the type of template to be invoked:

```
NLG(ReplyLocation,@IS)
```

A simplified version of the templates needed for this example are the following:

```
Template : ReplyLocation {
    cltype = decl;
     v_pred = EXIST_pres_3sg;
    obj.head = DevSpec.DevType.CONT;
    obj.quant = RefRes.QUANT;
    @if ( RefRes.QUANT > 1)
        @then {  obj.agr.num = plur; }
        @else {  obj.agr.num = sing; }
    obj.pmod =
       ApplyTemplate(LocativePP,DevSpec.Loc);
    obj.mod.a_pred = DevSpec.State.CONT;
    obj.mod.agr =  obj.agr;
}

Template : LocativePP {
    def = yes;
    head = CONT;
    loc = yes;
}
```

The templates above allow for the generation of different sentence patterns, depending on how much information has been supplied by the user. Thus, four variations could be generated from the same template:

- EXIST N Device
- EXIST N Device LOCATION
- EXIST N Device STATE
- EXIST N Device STATE LOCATION

After template application, the following deep syntactic feature structure is obtained:

$$\left[\begin{array}{ll} \text{cltype} & \text{decl} \\ \text{v\_pred} & \text{EXIST\_pres\_3sg} \\ \text{obj} & \left[\begin{array}{ll} \text{head} & \text{LIGHT}\\ \text{quant} & 2 \\ \text{agr} & \left[\begin{array}{ll}\text{num} & \text{plur}\end{array}\right] \\ \text{pmod} & \left[\begin{array}{ll} \text{head} & \text{KITCHEN}\\ \text{def} & \text{yes}\\ \text{loc} & \text{yes}\end{array}\right]\end{array}\right] \\ \text{mod} & \left[\begin{array}{ll}\text{a\_pred} & \text{ON}\\ \text{agr} & \left[\begin{array}{ll}\text{num} & \text{plur}\end{array}\right]\end{array}\right]\end{array}\right]$$

It should be pointed out that the representation obtained is language–independent at this point.

4 Linguistic Realization

As explained above, the process of linguistic realization is carried out by pre–existing transfer and generation modules from Episteme's MT engine (Amores and Quesada, 1997).

We have subdivided the overall process in three stages:

1. Lexicalization
2. Syntactic generation
3. Morphological realization

4.1 Lexicalization

Lexicalization is in charge of choosing the appropriate lexical items depending on the register, context and domain. A lexical transfer module obtains the appropriate lexical items and agreement information for the abstract concepts in the structure above. It is therefore viewed as a translation process, which may be subject to different contextual restrictions in order to obtain the desired linguistic variation.

The following are a few lexical transfer rules into Spanish. Similar ones have been developed for English as well.

```
LexicalTransfer a_pred

(EXIST_pres_3sg => haber
   @do
    {  tense = pres;
       agr.num = sing;
       agr.per = 3; }

LexicalTransfer a_pred
    (ON => encendido)
    (OFF => apagado)

LexicalTransfer head
    (LIGHT => luz @do
          {agr.gen =a fem;})
    (KITCHEN => cocina @do
          {agr.gen =a fem;})

LexicalTransfer def
    (yes => spec:el)

LexicalTransfer loc
    (yes => pcase:en)
```

The syntactic representation obtained can then be passed on to the surface morphosyntactic realization module:

$$\left[\begin{array}{ll} \text{cltype} & \text{decl} \\ \text{v\_pred} & \text{haber} \\ \text{tense} & \text{pres} \\ \text{agr} & \left[\begin{array}{ll}\text{num} & \text{sing}\\ \text{per} & 3\end{array}\right] \\ \text{obj} & \left[\begin{array}{ll} \text{head} & \text{luz}\\ \text{quant} & 2 \\ \text{agr} & \left[\begin{array}{ll}\text{num} & \text{plur}\\ \text{gen} & \text{fem}\end{array}\right] \\ \text{pmod} & \left[\begin{array}{ll} \text{head} & \text{cocina}\\ \text{pcase} & \text{en}\\ \text{agr} & \left[\begin{array}{ll}\text{num} & \text{sing}\\ \text{gen} & \text{fem}\end{array}\right]\end{array}\right] \\ \text{mod} & \left[\begin{array}{ll}\text{a\_pred} & \text{encendido}\\ \text{agr} & \left[\begin{array}{ll}\text{num} & \text{plur}\\ \text{gen} & \text{fem}\end{array}\right]\end{array}\right]\end{array}\right]\end{array}\right]$$

4.2 Syntactic realization

The surface realization module takes as input a feature structure, and returns a tree as a result. In this phase we will make use of the generation grammar already available as part of the Episteme machinery.

As an illustration, the rule below is responsible of generating VP's which contain an object and a prepositional object.

```
GenerationBlock: pred obj pobj
    (4:VP -> VG NP PP)
        {@self-2 = @generate(@up.obj);
         @self-3 = @generate(@up.pobj);
         @self-1 = @generate(@up); }
```

4.3 Morphological Realization

Finally, the morphological module returns inflected forms of terminal symbols. Our Spanish lexicon contains more than 900,000 fully inflected forms, of which only a portion are really needed in the dialogue application.

The result obtained after all modules have been called is a sentence like *Hay dos luces encendidas en la cocina* (There are two lights on in the kitchen).

5 *Advantages and Limitations*

5.1 Template–based generation

There has been much debate as to what exactly is a template–based natural language generation system, as opposed to a real NLG system. According to (van Deemter, Krahmer, and Theune, 2005), "template–based systems are natural language generating systems that map their non–linguistic input *directly* (i.e. without intermediate representations) to the surface linguistic surface structure." Other template–based systems such as YAG, EXEMPLARS, TG/2 and D2S offer more than the kind of simple string manipulation which is usually associated with the term 'templates' (Reiter, 1995).

Our particular implementation also goes beyond what is usually expected of a template since several intermediate representations are obtained during the generation process. Such level of modularization is potentially more fruitful than having to define monolithic templates for each natural language expression.

5.2 Recursivity and choice of templates

One of the advantages of our approach is that templates incorporate the capacity of calling other templates, which ensures that fewer templates are created and some of them can be reused. In the current stage of the implementation the choice of template to be generated is specified in the PostActions of the dialogue rule. No attempt is made at this point to choose the best template from a set depending on the context, or partially combining two templates to generate a complex utterance.

5.3 Language Independence

Another advantage of our approach is that sentence planning is language independent. How each conceptual feature translates into different languages is specified in the lexical transfer module described above, which ensures that multilingual language generation is much simpler than having to develop independent templates for each target language.

5.4 Domain Independence

In the current implementation, some features are still domain–dependent. As discussed above, this level of domain–dependence should be expected during the sentence planning process. For instance, in the in–home domain at hand, a **DevSpec** feature describes the features of the devices, whereas in a different domain the name of that feature may vary (for example `DestSpec` in the case of the telephone operator).

Nevertheless, a possible solution for this would be the inclusion of wild–cards such as @ANY in the appropriate path, so that both possibilities could be satisfied.

```
Template : ReplyLocation {
    cltype = decl;
    v_pred = EXIST_pres_3sg;
    obj.head = @ANY.DevType.CONT;
    obj.quant = RefRes.QUANT;
    @if ( RefRes.QUANT > 1)
        @then { obj.agr.num = plur; }
        @else { obj.agr.num = sing; }
    obj.pmod =
       Template(LocativePP,DevSpec.Loc);
    obj.mod.a_pred = DevSpec.State.CONT;
    obj.mod.agr = obj.agr;
}
```

Thus, this template would be valid not only for sentences like *Hay 2 luces encendidas en el salón*, but also for *Hay 2 personas en la cocina*.

The next version of the system will incorporate this possibility.

5.5 Verbose vs Fragmentary generation

The default strategy in the current version follows dialogue alignment in dialogue. That is, short answers correspond to short commands, and viceversa. A more elaborate strategy is being explored as part of the multimodal turn planning and output realization in the overall multimodal setting.

6 *Linguistic Coverage and Future work*

The linguistic coverage of the current version corresponds to the patterns needed in the home domain within the Talk project:

telephone control and home devices control in Spanish.

In a later version we plan to attack a series of limitations in the current implementation, such as:

6.1 Aggregation strategies

Some type of aggregation strategy (Dalianis, 1999), (Reape and Mellish, 1999) would be especially useful in the home domain. Consider for example trying to reply to a highly general query to the system such as *How many devices are there in the house?*. A natural reply should organize the information in some way, either by giving priority to locations over types of devices (*In the kitchen there are, ..., in the living room ...*) or some other strategy.

6.2 Linguistic variation

Our degree of linguistic variation is currently restricted to generating several patterns from the same template. However, there is no strategy to generate different constituent orders, voice, etc.

6.3 SSML Markup

Finally, SSML markup (Burnett, Walker, and Hunt, 2004) will be implemented in the next version of the system.

References

Amores, J. G. and J. F. Quesada. 1997. Episteme. *Procesamiento del Lenguaje Natural*, (21):1–15.

Amores, J.G. and J.F. Quesada. 2001. Dialogue moves for natural command languages. *Procesamiento del Lenguaje Natural*, (27):89–96.

Baptist, Lauren and Stephanie Seneff. 2000. Genesis-ii: A versatile system for language generation in conversational system applications. In *6th International Conference on Spoken Language Processing, ICSLP*, Beijing, China, October.

Bateman, John A. 1997. Enabling technology for multilingual natural language generation: the kpml development. *Natural Language Engineering*, 3(1):15–55.

Burnett, Daniel C., Mark R. Walker, and Andrew Hunt. 2004. Speech synthesis markup language (ssml) version 1.0. Recommendation 7, W3C, September.

Buseman, Stephan and Helmut Horacek. 1998. A flexible shallow approach to text generation. In *Proceedings 9th International Workshop on Natural Language Generation*, pages 238–247, Canada.

Dalianis, H. 1999. Aggregation in natural language generation. *Computational Intelligence*, 15(4):384–414, November.

DeSmedt, Konrad, Helmut Horacek, and Michael Zock Zock. 1996. Architectures for natural language generation: Problems and perspectives. In Giovanni Adorni and Michael Zock Zock, editors, *Trends in Natural Language Generation: An Artificial Intelligence Perspective*, Springer Verlag lecture notes in artificial intelligence 1036). Springer, Berlin, pages 17–46.

Dhomme, Project. 2001. Dialogues in the home machine environment. 5th Framework Programme.

Elhadad, M. and J. Robin. 1997. Surge: A comprehensive plug–in syntactic realisation component for text generation. Technical report, Computer Science Department, Ben–Gurion University, Beer Sheva, Israel.

Lavoie, B. and O. Rambow. 1997. A fast and portable realizer for text generation. In *Proceedings of the 5th Conference on Applied Natural–Language Processing (ANLP-1997).*

McRoy, Susan W., Songsak Channarukul, and Syed S. Ali. 2003. An augmented template–based approach to text realization. *Natural Language Engineering*, 9(4):381–420.

Oh, A. H. and A. Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 27–32, May.

Rambow, Owen, Srinivas Bangalore, and Marilyn Walker. 2001. Natural language generation in dialog systems. In *Proceedings of the first international conference on Human language technology research*, San Diego.

Reape, M. and C. Mellish. 1999. Just what is aggregation anyway. In *Proceedings of the 7th European Workshop on Natural Language Processing*, pages 20–29, Tolouse, France, May.

Reiter, Ehud. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation (INLGW–1994)*, pages 163–170, Kennebunkport, Maine.

Reiter, Ehud. 1995. Nlg vs templates. In *Proceedings of the 5th European Workshop on Natural Language Generation (EWNLG'95)*, pages 95–106, Leiden, The Netherlands.

Reiter, Ehud and Robert Dale. 2000. *Building Natural Language Generation Systems.* Studies in Natural Language Processing. Cambridge University Press, Cambridge.

Siridus, Project. 2002. Specification, interaction and reconfiguration in dialogue understanding systems. 5th Framework Programme.

Stenzhorn, Holger. 2002. A natural language generation system using xml- and java–technologies. In *Proceedings 2nd Workshop on NLP and XML*, Taipei, Taiwan.

Talk, Project. 2004. Talk and look: Linguistic tools for ambient linguistic knowlege. 6th Framework Programme.

Theune, Mariët, Esther Klabbers, Jan-Roelof de Pijper, Emiel Krahmer, and Jan Odijk. 2001. From data to speech: a general approach. *Natural Language Engineering*, 7(1):47–86.

Traum, David R. and Staffan Larsson. 2001. The information state update approach to dialogue modelling. In *The Trindi Book. Notes from the ESSLI–2001*, pages 18–21.

van Deemter, Kees, Emiel Krahmer, and Mariët Theune. 2005. Real vs. template–based nlg: a false opposition? *Computational Linguistics*, 31(1):15–24.

van Deemter, Kees and Jan Odijk. 1997. Context modelling and the generation of spoken discourse. *Speech Communication*, 21(1/2):101–121.

White, Michael and Ted Caldwell. 1998. Exemplars: A practical, extensible framework for dynamic text generation. In *Proceedings 9th International Workshop on Natural Language Generation*, pages 266–275, Canada.

Zhong, Huayan. 2004. Stochastic template–based surface realization. Rpe report, Department of Computer Science. State University of New York at Stony Brook, March.