# Experiences with and lessons learned from working with a modular natural language dialogue architecture

*Nils Dahlbäck Arne Jönsson*

Department of Computer and Information Science
Linköping University
{nilda, arnjo}@ida.liu.se

## Abstract

We present our experiences with developing and adapting a modular dialogue architecture for Natural Language human computer dialogues. We advocate a design strategy where from the beginning issues of portability are addressed by making clear which aspects of a system that are meant to be portable and which that are to be customized. Portability issues concern both shifts to a new platform and shifts on the same platform to a new application. We argue that portability is not an either or-issue. A corollary of this view is that the system should be clearly modularized from the beginning. However, in our experience, it is not possible to do this simply by using pre-developed building blocks, instead other forms of re-use is required.

## 1 Introduction

Portability has been a long time dream for developers of NL-based interactive systems. From the developers' standpoint, the huge cost for the development of working NL-systems has been one driving factor behind this. From the users' perspective, the motivating factor has been to be able to access the same information in a natural way regardless of whether sitting in front of a powerful workstation or on the move and using only a hand-held device. However, while both the needs of the users and the developers point in the same direction, we are still far from achieving this goal. It is our belief that one essential pre-requisite for progress in this area is a deeper understanding of which aspects, and under which conditions, the systems can be ported to other platforms and/or domains. Furthermore, it is our belief that one way of gaining this deeper knowledge is to reflect on experiences obtained in moving whole or parts of NL-systems from one platform or domain to another. In this spirit, we will here present some of our practical experiences of this and some tentative conclusions possible to draw from them

## 2 LINLIN architecture and systems developed

Since the mid 80'ies we have investigated human-computer natural language interaction, both theoretically, mainly through Wizard of Oz-studies and practically through the development of a number of research systems utilising varying domains, interaction styles and platforms. Our initial investigations focused on a sub-set of potential human-computer dialogues in natural language, namely a restricted subset of information seeking dialogues (IR). The reason for this choice was not only that it was a promising domain from the point of the users' needs, but also that we found

that such dialogues have some important characteristics that set them apart from most other human dialogues. The most important of these are that (1) there is no need to understand the underlying reasons, motives or goals behind the asking of the question. For instance, an answer to a question on the fuel consumption of a Volvo 850 is not dependent on whether you are a prospective buyer or have made a bet with a friend on which of your cars has the best mileage. (2) There is no need for sophisticated domain reasoning to provide an answer to the question. Taken together, these two entail that (3) there is no need for an explicit dynamic user model containing information not presented explicitly in the dialogue, since the user's domain knowledge and goals are the two major components of user models.

The research also resulted in LINLIN, a dialogue systems framework developed to handle dialogues of the type discussed above (Jönsson, 1997). The dialogue model in LINLIN is structurally based and uses a dialogue grammar for dialogue control. Contrary to many other dialogue systems (cf. Alexandersson & Reithinger, 1995) decisions as to what to do next are made on the basis of how focal parameters have been specified. Focus information is modelled as Objects, which identify a set of primary referents, Properties which identify a complex predicate ascribed to this set, and Markers for various sub-classifications (OPM). To make LINLIN portable to information providing content domains, other than the ones we originally worked with, we made a clear-cut distinction between processing modules, which should be domain independent, and knowledge sources, which were to be customized for each application. LINLIN has in MALIN been further developed to handle multimodal interaction and more advanced domains. In MALIN some of our initial restrictions are not met, as discussed below. The most important of our systems are presented in table 1.

**Table 1:** Some systems developed and studied. Abbreviations are explained in the text

| | **Properties** | | | | | |
|---|---|---|---|---|---|---|
| **System** | *Status* | *Domain* | *Type* | *Modules* | *Platform* | *Users' interaction style* |
| *PUB* | WizOz | Library | IR | OPM | Computer | Typed dialouge |
| *Wines* | WizOz | Wine selections | Advisory | OPM | Computer | Typed dialogue |
| *Travel* | WizOz | Travel agency | IR<br>IR+buying | OPM<br>OPM+ISF | Computer | Typed dialogue + graphics |
| *Cars* | WizOz<br>Implemented | Consumer report information | IR | OPM | Computer | Typed dialogue + tables |
| *Ötraf* | WizOz<br>Implemented | Local bus traffic information | IR | OPM+ISF+DKM | Computer | Typed dialogue, graphics, point and click, speech |
| *NokiaTV* | Implemented | TV-programs | IR | OPM | TV set top box | Speech input, typed output |
| *BirdQuest* | Implemented | Bird information | IR<br>IR+Identify | OPM+DKM<br>OPM+ISF+DKM | Computer<br>PDA<br>Mobile phone | Typed dialogue, speech, graphics, hand written dialogue |

## 3 Some observations

We will here summarize some of our observations, both conclusions we have drawn and hypotheses that have emerged through our work with these systems. Portability to another domain of the same type is reasonably straight forward. However, it is in this context important to make a distinction between similarity in dialogue type and dialogue properties on the one hand, and domain properties on the other. Our basic dialogue model developed initially for the cars system could be used also for the other domains. This seems also to hold for other IR dialogues we have studied using Wizard of Oz-methods. But observations from a simulated advisory system (Wines) indicate that this may not to be the case for such systems.

Furthermore, while the dialogue manager and grammar could be re-used for all the IR-dialogues, this was not true for the entire system architecture. We see a further distinction from pure IR-dialogues to what Allen *et. al* (2001) term *frame dialogues* as seen in the systems Travel, Ötraf and BirdQuest. In these systems we also needed means for specifying compound request objects that could not be managed only by OPM:s. We needed a frame structure termed Information Specification Forms (ISF) (Dahlbäck & Jönsson, 1997). ISFs are used by the dialogue system to, in a frame-filling session, request information from the user, before information retrieval is possible, e.g. for booking, finding a bus route or identifying a bird from observed properties. The Ötraf and BirdQuest systems also required the development of a Domain Knowledge Manager or DKM (Flycht-Eriksson 2000) able to carry out sophisticated domain and ontological reasoning. ISF:s, OPM:s and the DKM are not pre-defined building blocks, but complex modules or frameworks which require customisation for each new application. However, they are incrementally added to the existing architecture and modules, which could be re-used from previous systems.

In the course of customising MALIN, three complementary forms of re-use from a modular framework have evolved (Degerstedt & Jönsson, 2001): (1) tools: customisation through well-defined parameter settings and data representation files, e.g. the use of a parser to process a rule-based knowledge source; (2) framework templates: framework templates and application-specific code are kept in separate code trees; (3) code patterns: sub-modules, edited code patterns and other forms of textual re-use. Tools are a strong form of re-use but often limited in scope and flexibility. The use of framework templates is typically a more complex process but offers more support for construction of a complete application. A code pattern is the weakest form of re-use. It yields an important prototypical code or data skeleton to work from. Much of the tools, frameworks, libraries and prototype systems reside on our Open Source site nlpFarm: http://nlpfarm.sourceforge.net/. Modi is one such framework for enabling natural language processing to allow clients over a wireless network. Modi can be used by any dialogue system; all it assumes is a socket connection to pass messages to and from a mobile phone or PDA running Sun Java Micro edition (MIDP).

Portability to other platforms for similar kinds of dialogues is also possible within our architecture. This, at least, holds in our experience when going from typed only dialogue (PUB), via typed dialogue with graphics (Travel), and spoken input with written responses (NokiaTV) to combined point-and-click and language dialogue (spoken or typed) (Ötraf). But note that for all these cases, the user has visual support for maintaining the connected dialogue. Either by having the previous dialogue utterances visually preserved, or by constantly viewing a representation of the domain in form of maps and other graphics. Or in some cases both of these. We believe that this supports the maintenance of a common ground (Clark, 1992). In the cases of smaller devices, such as PDA:s or mobile phones the situation is different. Our work with this has just started, so it is too early to draw any conclusions. We are, however, reasonably convinced that utterance length will be differ-

ent in these cases. We also currently entertain the hypothesis that the dialogue structure will be more complex, through the need for a continuous re-establishing of referential expressions. Whether this can be accommodated using the same basic dialogue grammar we have used previously is currently an open issue. But we believe that if extensions are needed, these will be possible to accommodate within the existing basic architecture.

Another important observation from this work was that users with different degrees of knowledge of the domain had different preferences for interaction modality (Qvarfordt & Jönsson, 1998). This shows that the separation between dialogue and domain knowledge modules in our architecture does not imply that they are functionally independent in the sense that differences in users' domain knowledge will only affect the structure and content of the domain knowledge module. Also the dialogue structure and interaction modality preferences may be affected by this.

# 4 Discussion

There are two analytically distinct aspects involved when adapting an existing dialogue system to another application; adaptation to a new platform or device, or moving to another content domain. Given this, we believe a fruitful approach to working with multi-platform systems and multi-domain systems is to work with a modular system architecture, general enough to be adaptable to the different kinds of systems considered (though not necessarily to all potential NL-dialogues or human dialogues). Furthermore, we believe in the benefits of using a system development method that fits the chosen architecture and domain of applications under consideration. In this paper we have presented some observations from our work within such a framework.

One striking observation form the systems we have developed is the fact that very small, if any, changes have been needed for the dialogue component (Dialogue Manager) of our system, when moving from one application to another, or from one platform to another. And this despite its obvious limitations as a general model of dialogue (cf. Levinson, 1981).

Another observation from our work is that the kind of modular architecture that we have worked with lends itself well to adaptation to new applications. But we have also seen that moving to a new domain might require new capabilities from the system that couldn't be foreseen in advance. One case of this is our system for local bus-traffic timetable information. When we started working on this, there were quite a few air traffic information systems (ATIS) under development. With hindsight the difference between asking for flight and bus information mentioned above, i.e. that people know the name of the city they want to go to, but not the name of the bus stop nearest to their destination in their home town, is obvious. But it was not to us when we started our work. These additions have been necessary since we have worked in domains where all our initial restrictions do not apply to the same extent. The DKM is the clearest example of this, since it is a case of domain reasoning.

On the other hand, the necessary additions could be made to the basic simple grammar-based dialogue manager, as separate modules in the architecture. We believe that this is due to the fact that within variations, all dialogue types we have worked with still belong to a restricted class of human-computer dialogues, though not quite as narrow as in our initial work. But what still remains true is that an understanding of the user's underlying goals or reasons for engaging in the dialogue is not necessary. Exactly how far this requirement can be met is not clear to us, but we hypothesize that it for instance does not hold for advisory systems, and we are uncertain whether it holds for tutoring systems.

Our experience also shows that within the restricted class of dialogues we have worked with, the mapping between dialogue type and required dialogue architecture is rather complex. For instance, ISF:s are needed in the travel ordering dialogues, but not in the pure IR-dialogues in the same domain. But they are needed in other IR-dialogues, such as in the Ötraf system. So there is no simple mapping between dialogue type and necessary architecture. Instead the requirements on the architecture are set by a complex interplay of properties of the domain and the dialogue type that we do not at present have a full grasp of.

Given these observations, we believe that a path worth taking for research on natural language multimodal and speech interaction, is to concomitantly work on two levels; incrementally develop systems from restricted to less restricted dialogue situations (Degerstedt & Jönsson, 2001), and gradually develop a framework or descriptive taxonomy for natural language dialogues with computers (Dahlbäck, 1997), and doing this in a close interaction between the two areas. By only working on system development, we will not be able to know to which other cases experiences gained, both positive and negative, apply. And similarly, a framework or taxonomy that does not build on the experiences of studying existing systems will not make distinctions of relevance for the particular field we are working in, but more be general theories of dialogues in general. We hope that we with the examples here both have demonstrated the fruitfulness of this approach, and have illustrated the need for much further work in the area.

## 5 Acknowledgements

We are indebted to the current and former members of the natural language processing laboratory. The research reported here was financed by Vinnova, HSFR, NUTEK, and KFB.

## 6 References

Alexandersson, J. & Reithinger, N. (1995). Designing the dialogue component in a speech translation system. *Proceedings of the Ninth Twente Workshop on Language Technology (TWLT-9),* pp: 35--43

Allen, J.F., Byron, D.K., Dzikovska, M., Ferguson, G., Galescu, L., Stent, A. (2001). Toward conversational human-computer interaction. AI Magazine, 22, pp. 27–37.

Clark, H. (1992) *Arenas of language use*. Chicago: University of Chicago Press.

Dahlbäck, N. (1997). Towards a dialogue taxonomy. In E. Maier, M. Mast, S. LuperFoy (Eds.) Dialogue processing in spoken language systems. SpringerVerlag, LNAI1236.

Degerstedt, L. & Jönsson, A. (2001). A method for systematic implementation of dialogue management. In Workshop notes from the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Seattle, WA.

Flycht-Eriksson, A. (2000). A Domain Knowledge Manager for Dialogue Systems. *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI 2000*, IOS Press, Amsterdam.

Jönsson, A. A Model for Habitable and Efficient Dialogue Management for Natural Language Interaction, Natural Language Engineering, Vol. 3, No. 2/3 (1997) 103-122.

Levinson, S. C. (1981). Some Pre-Observations on the Modeling of Dialogue, *Discourse Processes*, Vol: 4, pp: 93-116

Qvarfordt, P. & Jönsson, A. (1998). Effects of Using Speech in Timetable Information Systems for WWW, *Proceedings of ICSLP'98,* Sydney, Australia, pp. 1635-1639