Enhanced Monitoring Tools and Online Dialogue Optimisation Merged into a New Spoken Dialogue System Design Experience

Ghislain Putois
Orange Labs
Lannion, France

Romain Laroche
Orange Labs
Issy-les-Moulineaux, France

Philippe Bretier
Orange Labs
Lannion, France

firstname.surname@orange-ftgroup.com

Abstract

Building an industrial spoken dialogue system (SDS) requires several iterations of design, deployment, test, and evaluation phases. Most industrial SDS developers use a graphical tool to design dialogue strategies. They are critical to get good system performances, but their evaluation is not part of the design phase.

We propose integrating dialogue logs into the design tool so that developers can jointly monitor call flows and their associated Key Performance Indicators (KPI). It drastically shortens the complete development cycle, and offers a new design experience.

Orange Dialogue Design Studio (ODDS), our design tool, allows developers to design several alternatives and compare their relative performances. It helps the SDS developers to understand and analyse the user behaviour, with the assistance of a reinforcement learning algorithm. The SDS developers can thus confront the different KPI and control the further SDS choices by updating the call flow alternatives.

Index Terms : Dialogue Design, Online Learning, Spoken Dialogue Systems, Monitoring Tools

1 Introduction

Recent research in spoken dialogue systems (SDS) has called for a "synergistic convergence" between research and industry (Pieraccini and Huerta, 2005). This call for convergence concerns architectures, abstractions and methods from both communities. Under this motivation, several research orientations have been proposed. This paper discusses three of them : dialogue design, dialogue management, and dialogue evaluation. Dialogue design and dialogue management reflect in this paper the respective paths that industry and research have followed for building their SDS. Dialogue evaluation is a concern for both communities, but remains hard to put into operational perspectives.

The second Section presents the context and related research. The third Section is devoted to the presentation of the tools : the historical design tool, its adaptation to provide monitoring functionalities and the insertion of design alternatives. It is eventually concluded with an attempt to reassessing the dialogue evaluation. The fourth Section describes the learning integration to the tool, the constraints we impose to the learning technique and the synergy between the tools and the embedded learning capabilities. Finally, the last Section concludes the paper.

2 Context

The spoken dialogue industry is structured around the architecture of the well known industrial standard VoiceXML [1]. The underlying dialogue model of VoiceXML is a mapping of the simplistic turn-based linguistic model on the browser-server based Web architecture (McTear, 2004). The browser controls the speech engines (recognition and text-to-speech) integrated into the voice platform according to the VoiceXML document served by an application server. A VoiceXML document contains a set of prompts to play and the list of the possible interactions the user is supposed to have at each point of the dialogue. The SDS developers [2], reusing Web standards and technologies (e.g. J2EE, JSP, XML...), are used to designing directed dialogues modelled by finite state automata. Such controlled and deterministic development process allows the spoken

1. http ://www.w3c.org/TR/voicexml20/
2. In this paper, the term "SDS developers" denotes without any distinction VUI designers, application developers, and any industry engineers acting in SDS building.

dialogue industry to reach a balance between usability and cost (Paek, 2007). This paper argues that tools are facilitators that improve both the usability vs. cost trade-off and the reliability of new technologies.

Spoken dialogue research has developed various models and abstractions for dialogue management : rational agency (Sadek et al., 1997), Information State Update (Bos et al., 2003), functional models (Pieraccini et al., 2001), planning problem solving (Ferguson and Allen, 1998). Only a very small number of these concepts have been transferred to industry. Since the late 90's, the research has tackled the ambitious problem of automating the dialogue design (Lemon and Pietquin, 2007), aiming at both reducing the development cost and optimising the dialogue efficiency and robustness. Recently, criticisms (Paek and Pieraccini, 2008) have been formulated and novel approaches (Williams, 2008) have been proposed, both aiming at bridging the gap between research –focused on Markov-Decision-Process (Bellman, 1957) based dialogue management– and industry –focused on dialogue design process, model, and tools. This paper contributes to extend this effort. It addresses all these convergence questions together as a way for research and industry to reach a technological breakthrough.

Regarding the dialogue evaluation topic, Paek (Paek, 2007) has pointed out that while research has exerted attention about "how best to evaluate a dialogue system ?", the industry has focused on "how best to design dialogue systems ?". This paper unifies those two approaches by merging system and design evaluation in a single graphical tool. To our knowledge, ODDS is the only industrial tool which handles the complete system lifecycle, from design to evaluation.

The tools and methods presented below have been tested and validated during the design and implementation of a large real-world commercial system : the 1013+ service is the Spoken Dialogue System for landline troubleshooting for France. It receives millions of calls a year and schedules around $8,000$ appointments a week. When the user calls the system, she is presented with an open question asking her for the reason of her call. If her landline is out of service, the Spoken Dialogue System then performs some automated tests on the line, and if the problem is confirmed, try and schedule an appointment with the user for a manual intervention. If the system and the user cannot agree on an appointment slot, the call is transferred to a human operator.

3 The tools

Industry follows the VUI-completeness principle (Pieraccini and Huerta, 2005) : *"the behaviour of an application needs to be completely specified with respect to every possible situation that may arise during the interaction. No unpredictable user input should ever lead to unforeseeable behaviour"*. The SDS developers consider reliable the technologies, tools, and methodologies that help them to reach the VUI-completeness and to control it.

3.1 The Dialogue Design Tool

The graphical abstraction proposed by our dialogue design tool conforms to the general graph representation of finite state automata, with the difference that global and local variables enable to factorise several dialogue states in a single node. Transitions relate to user inputs or to internal application events such as conditions based on internal information from the current dialogue state, from the back-end, or from the dialogue history. In that sense, dialogue design in the industry generally covers more than strict dialogue management, since its specification may indicate the type of spoken utterance expected from the user at each stage of the dialogue, up to the precise speech recognition model and parameter values to use, and the generation of the system utterance, from natural language generation to speech synthesis or audio recordings.

Our dialogue design tool offers to the SDS developers a graphical abstraction of the dialogue logic, sometimes also named the call flow. Thanks to a dynamic VoiceXML generation functionality, our dialogue design tool brings the SDS developers the guarantee that VUI-completeness at the design level automatically implies a similar completeness at the implementation level. During maintenance, If the SDS developers modify a specific part of the dialogue design, the tool guarantees that solely the corresponding code is impacted. This guarantee impacts positively VUI-completeness, reliability, and development cost.

Figure 1 presents the design of a typical VoiceXML page. This page is used when the system asks the user to accept an appointment time

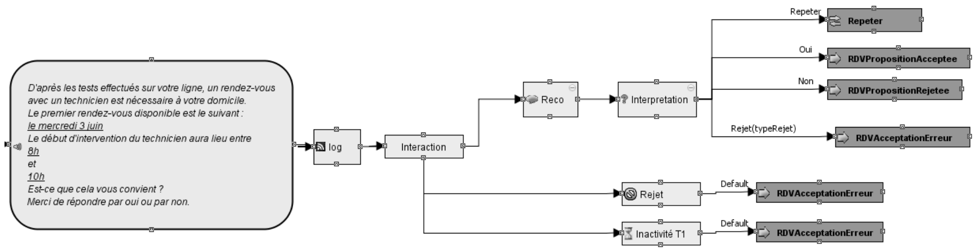

D'après les tests effectués sur votre ligne, un rendez-vous avec un technicien est nécessaire à votre domicile.
Le premier rendez-vous disponible est le suivant :
le mercredi 3 juin
Le début d'intervention du technicien aura lieu entre
8h
et
10h
Est-ce que cela vous convient ?
Merci de répondre par oui ou par non.

FIGURE 1 – 1013+ design excerpt : the system asks the user to confirm an appointment slot

slot. It first begins with a prompt box mixing static and dynamic prompts (the dynamic parts are underlined and realised by service-specific java code). A log box is then used some contextual session variables. Then, an interaction box is used to model the system reaction to the user behaviour : on the lower part of the Figure, we program the reaction to user inactivity or recognizer misunderstanding. In the upper part, we use a recognition box followed by a Natural Language Understanding (NLU), and we program the different output classes : repeat, yes, no and not understood. Each output is linked to a transition box, which indicates which VoiceXML page the service should call next.

3.2 Monitoring Functionalities inside the Design Tool

While researchers are focused on measuring the progress they incrementally reach, industry engineers have to deal with SDS tuning and upgrade. Their first dialogue evaluation KPI is task completion also called the automation rate because a SDS is deployed to automate specifically selected tasks. Most of the time, task completion is estimated thanks to the KPI. The KPI are difficult to exhaustively list and classify. Some are related to system measures, others are obtained thanks to dialogue annotations and the last ones are collected from users through questionnaires.

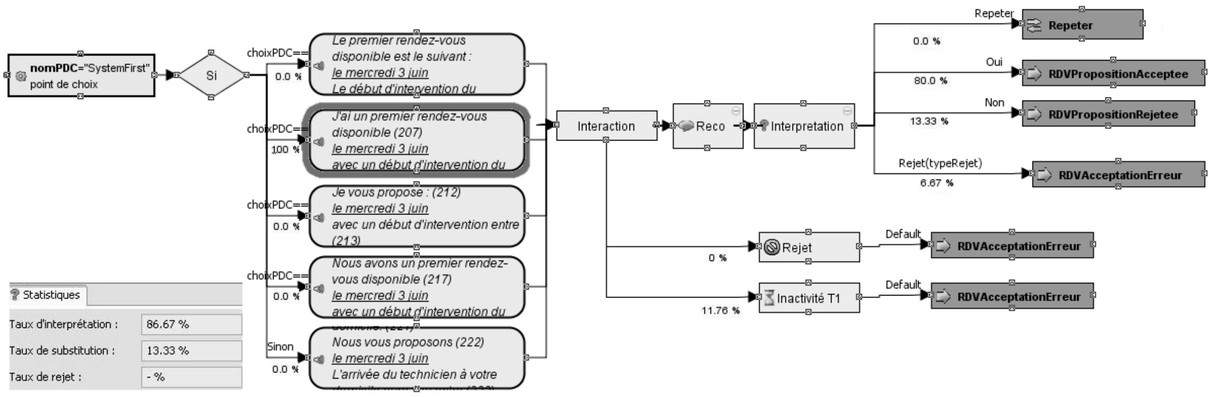
Some studies (Abella et al., 2004) investigated graphical monitoring tools. The corpus to visualise is a set of dialogue logs. The tool aims at revealing how the system transits between its possible states. As a dialogue system is too complex to enumerate all its possible states, the dialogue logs are regarded as a set of variables that evolve during time and the tool proposes to make a projection on a subset of these variables. This way, the generated graphs can either display the call flow, how the different steps are reached and where they lead, or display how different variables, as the number of errors evolve. This is mainly a tool for understanding how the users behave, because it has no direct connection with the way how the system was built. As consequence to this, it does not help to diagnose how to make it better. In other words, it does evaluate the system but does not meet one of our goal : the convergence between design and evaluation.

On the opposite, our graphical design tool provides an innovative functionality : local KPI projection into the original dialogue design thanks to an extensive logging. A large part of the KPI are automatically computed and displayed. As a consequence, it is possible to display percentage of which responses the system recognised, the users actually gave, and see how these numbers match the various KPI. It is one example among the numerous analysis views this graphical tool can provide.

3.3 Insertion of Alternatives

The 1013+ service has been used to test three kinds of design alternatives. The first kind is a strategy alternative : the service can choose between offering an appointment time slot to the client, or asking her for a time slot. This decision defines whether the next dialogue step will be system-initiative or user-initiative. The second kind is a speaking style alternative : the service can either be personified by using the "I" pronoun, adopt a corporate style by using the "We" pronoun, or speak in an impersonal style by using the passive mode. The third kind is a Text-To-Speech alternative : the service can use a different wording or prosody for a given sentence.

Figure 2 displays a monitoring view of an interaction implementation with alternatives. The recognition rate is the projected KPI on the graph at each branch. Other performance indicators are displayed at the bottom of the window : here, it


nomPDC="SystemFirst"
point de choix
Si
choixPDC==
0.0 %
Le premier rendez-vous disponible est le suivant : le mercredi 3 juin Le début d'intervention du
choixPDC==
100 %
J'ai un premier rendez-vous disponible (207) le mercredi 3 juin avec un début d'intervention du
choixPDC==
0.0 %
Je vous propose : (212) le mercredi 3 juin avec un début d'intervention entre (213)
choixPDC==
0.0 %
Nous avons un premier rendez-vous disponible (217) le mercredi 3 juin avec un début d'intervention du
Sinon
0.0 %
Nous vous proposons (222) le mercredi 3 juin L'arrivée du technicien à votre
Interaction
Reco
Interpretation
Repeter
0.0 %
Repeter
Oui
80.0 %
RDVPropositionAcceptee
Non
13.33 %
RDVPropositionRejetee
Rejet(typeRejet)
6.67 %
RDVAcceptationErreur
0 %
Rejet
Default
RDVAcceptationErreur
11.76 %
Inactivité T1
Default
RDVAcceptationErreur
Statistiques

Taux d'interprétation :	86.67 %
Taux de substitution :	13.33 %
Taux de rejet :	- %

FIGURE 2 – Some user experience feedbacks related to a selected prompt alternative.

is the actual rate of correct semantic decoding, the semantic substitution rate, and the semantic rejection rate. The selection of the highlighted box conditions the displayed logs.

Our design tool also provides a multivariate testing functionality. This method consists in testing multiple alternatives and selecting the best one on a fixed set of predetermined criteria. Regarding the VUI-completeness, presenting the complete automaton to the SDS developers is acceptable, as long as they can inspect and control every branch of the design. In general, they even come up with several competing designs or points of choice, which can only be properly selected from in a statistical manner. The ability to compare all the dialogue design alternatives in the same test-field is a major factor to boost up SDS enhancement by drastically reducing the time needed. When we were developing the current 1013+ version, we have been able to develop the 5 main alternatives in less than a month, where it had taken a month and a half for a unique alternative in previous versions. It brings a statistical relevance in the causal link between the tested alternatives and the differences in performance measures, because it ensures a good random input space coverage.

The KPI graphical projection into the dialogue design covers the dialogue alternatives : KPI computation just needs to be conditioned by the alternatives. Figure 2 illustrates the merge of several system prompt alternatives inside a single design. It represents the prompt alternatives the system can choose when proposing an appointment time slot. An action block informs the Learning Manager about the current dialogue state and available dialogue alternatives. An "If" block then activates the prompt alternative corresponding to a local variable "choixPDC" filled by the Learning Manager. The rest of the design is identical to the design presented in Figure 1.

The displayed KPI are conditioned by the selected alternative (here, the second wording circled in bold grey). ODDS then indicates how the dialogue call flow is breakdown into the different alternatives. As we have here conditioned the displayed information by the second alternative, this alternative receives 100% of the calls displayed, when the other alternatives are not used. We can then see the different outcomes for the selected alternative : the customer answer have lead to a timeout of the recognition in 11.78% of the cases, and amongst the recognised sentences, 80% were an agreement, 13.33% were a reject, and 6.67% were not understood.

On the bottom-left part, one can display more specific KPI, such as good interpretation rate, substitution rate, and reject rate. These KPI are computed after the collected logs have been manually annotated, which remains an indispensable process to monitor and improve the recognition and NLU quality, and thus the overall service quality.

Conditioning on another alternative would have immediately led to different results, and someway, embedding the user experience feedback inside the dialogue design forms a new material to touch and feel : the SDS developers can now sculpt a unique reactive material which contains the design and the KPI measures distribution. By looking at the influence of each alternative on the KPI when graphically selecting the alternatives, the SDS developers are given a reliable means to understand how to improve the system.

3.4 Reassessing Dialogue Evaluation

The traditional approaches to dialogue evaluation attempt to measure how best the SDS is adapted to the users. We remind that each interaction between the user and the SDS appears to be a unique performance. First, each new dialogue is co-built in a unique way according to both the person-specific abilities of the user and the possibilities of the SDS. Second, the user adapts very quickly to new situations and accordingly changes her practices. The traditional approaches to dialogue evaluation are eventually based on the fragile reference frame of the user, not reliable enough for a scientific and an industrial approach of the spoken dialogue field, mostly because of the inability to get statistical call volumes for all the dialogue alternatives.

This suggests for a shift in the reference frame used for dialogue evaluation : instead of trying to measure the adequacy between the SDS and the user in the user's reference frame, one can measure the adequacy between the user and the SDS in the design reference frame composed by the dialogue logic, the KPI and their expected values. Taking the design as the reference allows reassessing the dialogue evaluation. The proposed basis for dialogue evaluation is reliable for the SDS developers because it is both stable and entirely under control. Deviations from the predicted situations are directly translated into anomalous values of measurable KPI that raise alerts. These automatically computable alerts warn the SDS developers about the presence of issues in their dialogue design.

4 Dialogue design learning

As presented in previous Section, the alternative insertion is an enabler for the dialogue system analysis tools. It provides the SDS developers with a novel call flow visualisation experience. The further step to this approach is to automate at least a part of those analyses and improvements with learning capabilities.

4.1 Constraints

The objective is to automatically choose online the best alternative among those proposed in the design tool, and to report this choice to the SDS developers via the monitoring functionalities that are integrated to the design tool. This approach differs from the classical reinforcement learning methods used in the dialogue literature, which make their decisions at the dialogue turn level.

We use a technique from a previous work (Laroche et al., 2009). It does not need to declare the reachable states : they are automatically created when reached. This is also a parameter-free algorithm, which is very important when we consider that most dialogue application developers are not familiar with reinforcement learning theory. We keep the developer focussed on its main task. The two additional tasks required for the reinforcement learning are to define the variable set on which the alternative choice should depend, and to implement a reward function based on the expected evaluation of the task completion, in order to get a fully automated optimisation with an online evaluation. The dialogue system automatic evaluation is a large problem that goes beyond the scope of this paper. However, sometimes, the dialogue application enables to have an explicit validation from the user. For instance, in an appointment scheduling application, the user is required to explicitly confirm the schedule he was proposed. This user performative act completes the task and provides a reliable automatic evaluation.

4.2 Learning and Monitoring Synergy in the Design Optimisation

The learning algorithm and the SDS developers are two actors on the same object : the dialogue system. But, they work at a different time space. The learning algorithm updates its policy after each dialogue while the SDS developers monitor the system behaviour more occasionally. The same kind of opposition can be made on the action space of those actors. The learning algorithm can only change its policy among a limited amount of alternatives, while the SDS developers can make deeper changes, such as implementing a new dialogue branch, adding new alternatives, new alternative points, removing alternatives, etc...

Last but not least, their sight ranges vary a lot too. The learning algorithm is concentrated on the alternative sets and automatic evaluation and ignores the rest, while the SDS developers can apprehend the dialogue application as a whole, as a system or as a service. They can also have access to additional evaluations through annotations, or user subjective evaluations.

These functionality differences make their respective roles complementary. The SDS develop-

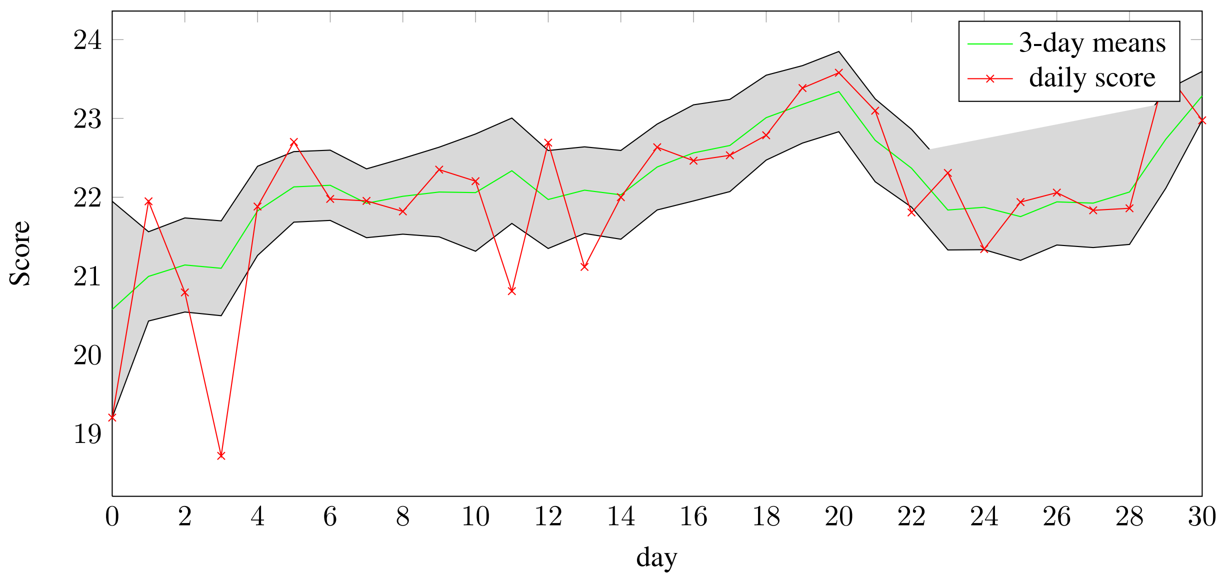


FIGURE 3 – Evolution of the system score

ers have the responsibility for the whole application and the macro-strategic changes while the learning manager holds the real-time optimisation.

4.3 Control vs. Automation : the Trusting Threshold

As argued by Pieraccini and Huerta (Pieraccini and Huerta, 2005), finite state machine applied to dialogue management does not restrict the dialogue model to strictly directed dialogues. Finite state machines are easily extensible to powerful and flexible dialogue models. Our dialogue design tool offers various extensions : dialogue modules, hierarchical design, arbitrary function invocation at any point of the design, conditional statements to split the flow in different paths. All those extensions allow designing any topology of the finite state machine required to handle complex dialogue models like mixed-initiative interaction. Dialogue model is not the point where research and industry fail to converge.

The divergence point concerns the control aspect of VUI-completeness versus the automation of the dialogue design. As pointed out by recent works (Paek and Pieraccini, 2008), MDP-based dialogue management aiming at automating the whole dialogue design is rejected by the SDS developers. Even more adaptive, it is seen as an uncontrollable black box sensitive to the tuning process. The SDS developers do not rely on systems that dynamically build their dialogue logic without a sufficient degree of monitoring and control.

Williams (Williams, 2008) has made a substantial effort to meet this industrial requirement. His system is a hybridisation of a conventional dialogue system following an industrial process, with a POMDP decision module, which is a MDP-based approach to dialogue management enhanced with dialogue state abstractions to model uncertainties. The responsibilities of each part of the system are shared as follows : the conventional system elects several candidate dialogue moves and the POMDP decision module selects the most competitive one. This is a great step towards industry because the dialogue move chosen by the POMDP module has been first controlled by the conventional system design. Nevertheless, the so-built hybrid system is still not fully compliant with the industrial constraints for the following reasons.

First, contrary to our approach, the SDS developer is called upon specific skills that cannot be demanded to a developer (modeling and tuning a (PO)MDP). This is a no-go for further integration in an industrial process.

Second, such a predictive module is not self-explanatory. Although the SDS developers have the control on the possible behaviour presented to the POMDP decision module, they are given no clue to understand how the choices are made. In fact, a learnt feature can never be exported to another context. At the opposite, our approach allows us to learn at the design level and consequently to report in the automaton the optimisation. The learning results are therefore understand-able, analysable and replicable on a larger scale, in a way similar to classical ergonomics guidelines (but statistically proved).

4.4 Learning results on the 1013+ service

In the 1013+ service, our experiments have focused on the appointment scheduling domain. We have chosen to integrate the following rewards in the service : each time a user successfully manages to get an appointment, the system is given a $+30$ reward. If the system is unable to provide an appointment, but manages to transfer the user to a human operator, the system is given a $+10$ (a "resit"). Last, if the user hangs up, the system is not given any positive reward. Every time the system does not hear nor understand the user, it is given a penalty of 1.

In the beginning of the experiment, when the system is still using a random policy, the completion rate is as low as 51%, and the transfer rate is around 36%. When the system has learned its optimal policy, the completion rate raises up to 70%, with a transfer rate around 20%. In our experiment, the system has learned to favour an impersonal speaking style (passive mode) and it prefers proposing appointment time slots rather than asking the user to make a proposition (the later case leading to lot of "in private" user talks and hesitations, and worse recognition performance).

Figure 3 shows the evolution of the mean dialogue score during the first month. Each server have its own Learning Manager database, and optimises separately. This is a welcome feature, as each server can address a different part of the user population, which is a frequent operational requirement.

The dialogue score drawn on Figure 3 is computed by averaging the mean dialogue score per server. The crossed line represents the daily mean dialogue score. The normal line represents the 3-day smoothed dialogue mean score. The grayed area represents the 95% confidence interval. During this first month of commercial exploitation, one can notice two major trends : at first, the dialogue score is gradually increasing until day 20, then the performances noticeably drops, before rising up again. It turns out that new servers were introduced on day 20, which had to learn the optimal dialogue policy. Ultimately (on the second month), they converge to the same solution as the first servers.

5 Conclusion

5.1 A New Basis for Trusting Automatic Learning

This paper presents an original dialogue design tool that mixes dialogue design and dialogue evaluation in the same graphical interface. The design paradigm supported by the tool leads the SDS developers to predict value ranges of local KPI while designing the dialogue logic. It results a new evaluation paradigm using the system design as the reference and trying to measure deviations between the predicted and the measured values of the designed local KPI. The SDS developers rely on the tool to fulfil the VUI-completeness principle. Classically applied to dialogue design, the tool enables its application to the dialogue evaluation, leading to the comparison of dialogue design alternatives.

This places the SDS developers in a dialogue design improvement cycle close to the reinforcement learning decision process. Moreover, the inspector offered by the user experience feedback functionality allows the SDS developers to understand, analyse and generalize all the decisions among the dialogue design alternatives. Combining the learning framework and the design tool guarantees the SDS developers keep control of the system. It preserves VUI-completeness and opens the way to a reliable learning based dialogue management.

5.2 Implementation

This approach to learning led us to deploy in October 2009 the first commercial spoken dialogue system with online learning. The system's task is to schedule an appointment between the customer and a technician. This service receives approximately $8,000$ calls every month. At the time those lines are written, we are already in a virtuous circle of removing low-rated alternatives and replacing them with new ones, based on what the system learnt and what the designer understands from the data.

5.3 Future Work

On a social studies side, we are interested in collaborations to test advanced dialogue strategies and/or information presentation via generation. Indeed, we consider our system as a good opportunity for large scope experiments.

6 Acknowledgements

This research has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 216594 (CLASSIC project : www.classic-project.org).

References

A. Abella, J.H. Wright, and A.L. Gorin. 2004. Dialog trajectory analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 441–444, May.

R.E. Bellman. 1957. A markovian decision process. *Journal of Mathematics and Mechanics*, 6 :679–684.

J. Bos, E. Klein, O. Lemon, and T. Oka. 2003. Dipper : Description and formalisation of an information-state update dialogue system architecture.

George Ferguson and James F. Allen. 1998. Trips : An integrated intelligent problem-solving assistant. In *In Proc. 15th Nat. Conf. AI*, pages 567–572. AAAI Press.

R. Laroche, G. Putois, P. Bretier, and B. Bouchon-Meunier. 2009. Hybridisation of expertise and reinforcement learning in dialogue systems. In *Proceedings of Interspeech. Special Session : Machine Learning for Adaptivity in Spoken Dialogue*, Brighton (United Knigdom), September.

O. Lemon and O. Pietquin. 2007. Machine learning for spoken dialogue systems. In *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, pages 2685–2688, August.

M. F. McTear. 2004. *Spoken Dialogue Technology : Toward the Conversational User Interface*. Springer, August.

T. Paek and R. Pieraccini. 2008. Automating spoken dialogue management design using machine learning : An industry perspective. *Speech Communication*, 50 :716–729.

T. Paek. 2007. Toward evaluation that leads to best practices : Reconciling dialog evaluation in research and industry. In *Proceedings of the Workshop on Bridging the Gap : Academic and Industrial Research in Dialog Technologies*, pages 40–47, Rochester, NY, April. Association for Computational Linguistics.

R. Pieraccini and J. Huerta. 2005. Where do we go from here ? research and commercial spoken dialog systems. In Laila Dybkjaer and Wolfgang Minker, editors, *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 1–10.

R. Pieraccini, S. Caskey, K. Dayanidhi, B. Carpenter, and M. Phillips. 2001. Etude, a recursive dialog manager with embedded user interface patterns. In *Automatic Speech Recognition and Understanding, 2001 IEEE Workshop on*, pages 244–247.

M. D. Sadek, P. Bretier, and F. Panaget. 1997. Artimis : Natural dialogue meets rational agency. In *in Proceedings of IJCAI-97*, pages 1030–1035. Morgan Kaufmann.

J. D. Williams. 2008. The best of both worlds : Unifying conventional dialog systems and POMDPs. In *International Conference on Speech and Language Processing*.