Combining Incremental Language Generation and Incremental Speech Synthesis for Adaptive Information Presentation

Hendrik Buschmeier[1], **Timo Baumann**[3], **Benjamin Dosch**, **Stefan Kopp**[1], **David Schlangen**[2]
[1]Sociable Agents Group, CITEC and Faculty of Technology, Bielefeld University
[2]Dialogue Systems Group, Faculty of Linguistics and Literary Studies, Bielefeld University
{hbuschme,bdosch,skopp,david.schlangen}@uni-bielefeld.de
[3]Natural Language Systems Division, Department of Informatics, University of Hamburg
baumann@informatik.uni-hamburg.de

Abstract

Participants in a conversation are normally receptive to their surroundings and their interlocutors, even while they are speaking and can, if necessary, adapt their ongoing utterance. Typical dialogue systems are not receptive and cannot adapt while uttering. We present combinable components for incremental natural language generation and incremental speech synthesis and demonstrate the flexibility they can achieve with an example system that adapts to a listener's acoustic understanding problems by pausing, repeating and possibly rephrasing problematic parts of an utterance. In an evaluation, this system was rated as significantly more natural than two systems representing the current state of the art that either ignore the interrupting event or just pause; it also has a lower response time.

1 Introduction

Current spoken dialogue systems often produce prescripted system utterances or use templates with variable substitution during language generation. If a dialogue system uses grammar-based generation at all, it produces complete utterances that are then synthesised and realised in one big chunk. As systems become increasingly more conversational, however, the need arises to make output generation[1] more flexible. In particular, capabilities for *incrementally* generating output become desirable, for two kinds of reasons.

(a) In situations where fast system responses are important, production of output can begin before the content that is to be presented is fully specified – even if what is being produced is just a turn-taking signal (Skantze and Hjalmarsson, 2010).

(b) A system that produces its output incrementally can react to events happening while it is realising an utterance. This can be beneficial in domains where the state of the world that the system relays information about can change mid-utterance, so that a need may arise to adapt while speaking. It should also improve naturalness by allowing the system to react to dialogue phenomena such as concurrent feedback signals from the user (Buschmeier and Kopp, 2011).

We present work towards enabling such capabilities. We have implemented and connected a component for incremental natural language generation (iNLG) that works with specifications of sub-utterance-sized communicative intentions and a component for incremental speech synthesis (iSS) that can handle sub-utterance-sized input and modifications to not-yet-spoken parts of the utterance with very low latencies. To explore whether such an output generation capability can indeed be advantageous, we have created a test system that can react to random noise events that occur during a system utterance by repeating and modifying the last sub-utterance chunk. In an evaluation, we found that this system is in general more reactive than a non-incremental variant and that humans rate its behaviour to be more natural than two non-incremental and non-responsive systems.

2 Related Work

Psycholinguistic research has identified incrementality as an important property of human language production early on and it has been incorporated into several models (e. g., Kempen and Hoenkamp, 1987;

[1]We will use the term 'output generation' here to cover both natural language generation and speech synthesis.

Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), pages 295–303,
Seoul, South Korea, 5-6 July 2012.

Levelt, 1989). Guhe (2007) presents a computational model of incremental conceptualisation. However, work on iNLG itself is rare, partly because NLG research focusses on text (instead of spoken language).

Notable exceptions are the in-depth analysis of requirements for and properties of incremental generation by Kilger and Finkler (1995), who describe the LTAG-based incremental syntactic generator VM-GEN. It takes incremental input, processes it and produces output as soon as at least a prefix of the final sentence is syntactically complete. If VM-GEN notices that it committed itself to a prefix too early, it can initiate an overt repair. More recently, Skantze and Hjalmarsson (2010) presented a system that performs incremental generation in the context of a spoken dialogue system. It can already start to produce output when the user has not yet finished speaking and only a preliminary interpretation exists. By flexibly changing what to say and by being able to make self-repairs the system can recover from situations where it selected and committed on an inadequate speech plan. Both systems, however, are not able to flexibly adapt the language that they generate to changing requirements due to changes in the situation or changing needs on the side of the user.

Real-time on-the-fly control of speech synthesis is rare, especially the full integration into a dialogue system. Matsuyama et al. (2010) describe a system that feeds back to the dialogue system the word at which it has been interrupted by a barge-in. Edlund (2008) additionally enables a system to continue at the point where it was interrupted. He also outlines some requirements for incremental speech synthesis: to *give constant feedback* about what has been delivered, to *be interruptible* (and possibly continue from that position), and to *run in real time*. Edlund's system, which uses diphone synthesis, performed non-incrementally before delivery starts. We go beyond this in also enabling changes during delivery and conducting synthesis steps *just-in-time*.

Dutoit et al. (2011) present an incremental HMM optimiser which allows to change pitch and tempo of upcoming phonemes. However, as that system is fed from a (non-incrementally produced) label file, it cannot easily be used in an incremental system.

A predecessor of our iSS component (which was not yet fully incremental on the HMM level) is described in detail in (Baumann and Schlangen, 2012a).

3 Incremental and Adaptive NLG

3.1 The SPUD microplanning framework

The NLG component presented here is based on the SPUD microplanning framework (Stone et al., 2003) and realised in DeVault's (2008) implementation 'Java SPUD'. SPUD frames microplannig as a constraint satisfaction problem, solving the tasks that are involved in generating a sentence (lexical and syntactic choice, referring expression generation and aggregation) in an integrated manner. Generation starts from a communicative goal that specifies constraints for the final utterance. The generation process is further shaped by (a) general constraints that model pragmatic properties of language use such as the Gricean maxims (a principle called 'textual economy'); (b) specific constraints imposed through the communicative status of the propositions to be communicated (i. e., what knowledge can be presupposed and what needs to be communicated explicitly); and (c) linguistic resources (a context-free tree rewriting formalism based on LTAG; Stone, 2002).

To deal efficiently with the infinite search space spanned by the linguistic resources, SPUD uses a heuristic search algorithm to find an utterance that satisfies the imposed constraints (Stone et al., [2003] describe the heuristic function). In each search step, the algorithm expands the 'provisional' utterance by adding the linguistic resource that maximally reduces the estimated distance to the final utterance.

If the generation process runs into a dead-end state, it could in principle deal with the situation by tracking back and expanding a different branch. This, however, is impractical, as it becomes impossible to project when – if at all – generation will finish. Hence, in that case, SPUD stops without providing a result, delegating the problem back to the preceding component in the generation pipeline.

3.2 *Partially* incremental generation

SPUD generates utterances incrementally in the sense that the completeness of the provisional utterance increases monotonically with every step. This, however, does not mean that the surface structure of provisional utterances is constructed incrementally (i. e., from left to right) as well, which would only be possible, if (a) the search strategy would always expand the leftmost non-lexicalised node in the provisional

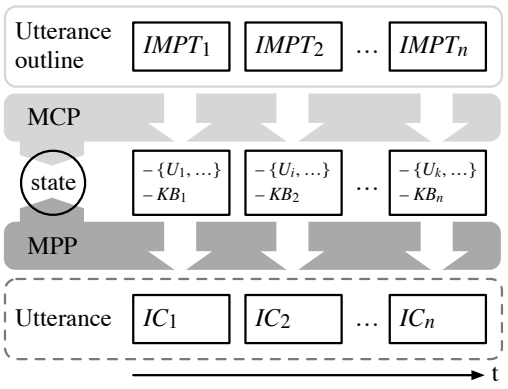

Figure 1: Incremental microplanning consists of two processes, micro content planning (MCP) and microplanning-proper (MPP). The former provides incremental microplanning tasks from an utterance outline to the latter, which incrementally transforms them into communicative intent and intonation unit-sized chunks of natural language.

utterance first and if (b) the linguistic resources are specified (and ordered) in a way that allows left-to-right expansion of the trees in all possible situations. In practice, both requirements are difficult to meet and full word-by-word incrementality in natural language microplanning is not within reach in the SPUD framework. Because of this, we take a slightly more coarse grained approach to incremental microplanning and choose chunks of the size of intonation phrases instead of words as our incremental units. We say that our microplanner does *'partially* incremental generation'.

Our incremental microplanner comprises two interacting processes, *micro content planning* and *microplanning-proper* (MCP and MPP; schematised in Figure 1), each of which fulfils a distinct task and operates on different structures.

MCP takes as input *utterance outlines* that describe the communicative goal (a set of desired updates U_x) intended to be communicated in an utterance and the presuppositions and private knowledge needed to do so. Importantly, utterance outlines specify how the communicative goal can be decomposed into an ordered list of incremental microplanning-tasks $IMPT_x$. Each such task comprises (a) a subset of the communicative goal's desired updates that belong together and fit into one intonation unit sized chunk of speech and (b) knowledge KB_x used in generation.

MPP takes one incremental microplanning-task at a time and uses SPUD to generate the IMPT's communicative intent as well as its linguistic surface form IC_x. The communiciative intent is added to a representation ('state' in Figure 1) that is shared between the two processes. While processing the IMPTs of an utterance outline, MCP can access this representation, which holds information about all the desired updates that were achieved before, and thus knows that a desired update that is shared between subsequent IMPTs has already been communicated. MPP can also take this information into account during generation. This makes it possible that an utterance is coherent and adheres to pragmatic principles even though generation can only take local decisions.

3.3 Adaptive generation

Being able to generate language in sub-utterance chunks makes it possible to dynamically adapt later increments of an utterance to changes in the situation that occur while the utterance is being realised. Decisions about these adaptations need not be taken almost until the preceding increment finishes, making the generation process very responsive. This is important to be able to deal with interactive dialogue phenomena, such as communicative feedback of the interlocutor (Allwood et al., 1992) or compound contributions (Howes et al., 2011), in a timely manner.

Adaptation may happen in both parts of incremental microplanning. In MCP, adaptation takes the form of dynamically changing the choice of which IMPT to generate next or changing the internal structure of an IMPT; adaptation in MPP changes the choices the generation process makes while transforming IMPTs into communicative intent and surface form. Adaptation in MCP is triggered top-down, by higher-level processes such as dialogue management. Adaptation in MPP on the other hand depends on the task given and on the status of the knowledge used during generation. The details are then governed by global parameter settings MPP uses during generation.

If there is, for example, reason for the system to believe that the current increment was not communicated clearly because of noise in the transmission channel, the MCP process might delay future IMPTs and initiate a repair of the current one by re-inserting it at the beginning of the list of upcoming IMPTs of this utterance outline. The MPP process' next task is then to re-generate the same IMPT again. Due to

Table 1: Surface forms generated from the same IMPT (desired updates = {hasSubject(event6, 'Vorlesung Linguistik')}; KB = {event6}) but with different levels of verbosity.

Verbosity	Generated sub-utterance chunk
0	'Vorlesung Linguistik' *(lecture Linguistcs)*
1	'Betreff: Vorlesung Linguistik' *(subject: lecture Linguistics)*
2	'mit dem Betreff Vorlesung Linguistik' *(with the subject: lecture Linguistics)*

changes in the state information and situation that influence microplanning, the resulting communicative intent and surface form might then differ from its previous result.

3.4 Adaptation mechanisms

As a proof of concept, we integrated several adaptation mechanism into our NLG-microplanning system. The goal of these mechanisms is to respond to a dialogue partner's changing abilities to perceive and/or understand the information the system wants to convey. Some of the mechanisms operate on the level of MCP, others on the level of MPP. The mechanisms are implemented either with the knowledge and its conversational status used in generation or by altering the decision structure of SPUD's search algorithm's heuristic function. Similar to the approach of flexible NLG described by Walker et al. (2007), most of the mechanism are conditioned upon individual flags, that in our case depend on a numeric value that represents the level of understanding the system attributes to the user. Here we describe the two most relevant mechanisms used to adapt verbosity and redundancy.

Verbosity The first mechanism aims at influencing the length of a sub-utterance chunk by making it either more or less verbose. The idea is that actual language use of human speakers seldom adheres to the idealised principle of textual economy. This is not only the case for reasons of cognitive constraints during speech production, but also because words and phrases that do not contribute much to an utterance's semantics can serve a function, for example by drawing attention to specific aspects of an utterance or by giving the listener time to process.

To be able to vary utterance verbosity, we annotated the linguistic resources in our system with values of their verbosity (these are hand-crafted similar to the rule's annotation with production costs). During generation in MPP the values of all linguistic resources used in a (provisional) utterance are added up and used as one factor in SPUD's heuristic function. When comparing two provisional utterances that only deviate in their verbosity value, the one that is nearer to a requested verbosity level is chosen. Depending on this level, more or less verbose constructions are chosen and it is decided whether sub-utterance chunks are enriched with additional words. Table 1 shows the sub-utterance chunk 'Betreff: Vorlesung Linguistik' *(subject: lecture Linguistics)* generated with different levels of verbosity.

Redundancy The second adaptation mechanism is redundancy. Again, redundancy is something that an ideal utterance does not contain and by design SPUD penalises the use of redundancy in its heuristic function. Two provisional utterances being equal, the one exhibiting less redundancy is normally preferred. But similar to verbosity, redundancy serves communicative functions in actual language use. It can highlight important information, it can increase the probability of the message being understood (Reiter and Sripada, 2002) and it is often used to repair misunderstanding (Baker et al., 2008).

In incremental microplanning, redundant information can be present both within one sub-utterance chunk (e. g., 'tomorrow, March 26, . . . ' vs. 'tomorrow . . . ') or across IMPTs. For the former case, we modified SPUD's search heuristic in order to conditionally either prefer an utterance that contains redundant information or an utterance that only contains what is absolutely necessary. In the latter case, redundancy only becomes an option when later IMPTs enable the choice of repeating information previously conveyed and therefore already established as shared knowledge. This is controlled via the internal structure of an IMPT and thus decided on the level of MCP.

4 Incremental Speech Synthesis

In this section we describe our component for incremental speech synthesis. We extend Edlund's (2008) requirements specification cited in Section 2, requiring additionally that an iSS supports changes to as-yet

unspoken parts of an ongoing utterance.

We believe that the iSS's requirements of interruptability, changeability, responsiveness, and feedback are best resolved by a processing paradigm in which processing takes place *just-in-time*, i. e., taking processing steps as late as possible such as to avoid re-processing if assumptions change. Before we describe these ideas in detail, we give a short background on speech synthesis in general.

4.1 Background on speech synthesis

Text-to-speech (TTS) synthesis functions in a top-down processing approach, starting on the utterance level and descending onto words and phonemes, in order to make good decisions (Taylor, 2009). For example, top-down modelling is necessary to assign stress patterns and sentence-level intonation which ultimately lead to pitch and duration contours, and to model co-articulation effects.

TTS systems start out assigning intonation patterns to the utterance's words and then generate a *target phoneme sequence* which is annotated with the targets' durations and pitch contour; all of this is called the linguistic pre-processing step. The synthesis step proper can be executed in one of several ways with HMM-based and unit-selection synthesis currently producing the perceptually best results.

In HMM-based synthesis, the target sequence is first turned into a sequence of HMM states. A global optimisation then determines a stream of vocoding features that optimise both HMM emission probabilities and continuity constraints (Tokuda et al., 2000). The stream may also be enhanced to consider global variance of features (Toda and Tokuda, 2007). The parameter frames are then fed to a vocoder which generates the final speech audio signal.

Unit-selection, in contrast, searches for the best sequence of (variably sized) units of speech in a large, annotated corpus, aiming to find a sequence that closely matches the target sequence while having few and if possible smooth joints between units.

We follow the HMM-based approach for our component for the following reasons: (a) even though only global optimisation is optimal for both techniques, the influence of look-ahead on the continuity constraints of HMM-based synthesis is linear leading to a linear loss in optimality with smaller look-aheads (whereas unit-selection with limited look-ahead may jump erratically between completely different unit sequences). (b) HMM-based synthesis nicely separates the production of vocoding parameter frames from the production of the speech audio signal which allows for fine-grained concurrent processing (see next subsection). (c) Parameters in the vocoding frames are partially independent. This allows us to independently manipulate, e. g., pitch without altering other parameters or deteriorating speech quality (in unit-selection, a completely different unit sequence might become optimal even for slight changes of pitch).

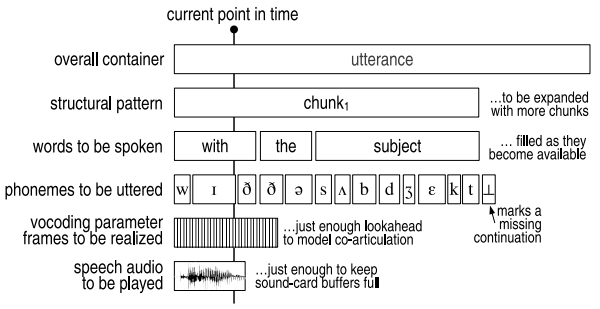

Figure 2: Hierarchical structure of incremental units describing an example utterance as it is being produced during delivery.

4.2 Incrementalising speech synthesis

As explained in the previous subsection, speech synthesis is performed top-down, starting at the utterance and progressing down to the word, target and finally, in the HMM approach, vocoding parameter and signal processing levels. It is, however, not necessary that all details at one level of processing are worked out before starting to process at the next lower level. To be precise, some syntactic structure is sufficient to produce sentence-level intonation, but all words need not be known. Likewise, post-lexical phonological processes can be computed as long as a local context of one word is available and vocoding parameter computation (which must model co-articulation effects) should in turn be satisfied with about one phoneme of context. Vocoding itself does not need any lookahead at all (aside from audio buffering considerations).

Thus, we generate our data structures incrementally in a top-down and left-to-right fashion with different amounts of pre-planning and we do this using several processing modules that work concurrently. This results in a 'triangular' structure as shown in

Figure 2. At the top stands a pragmatic plan for the full utterance from which a syntactic plan can be devised. This plan is filled with words, as they become available. On the vocoding parameter level, only a few frames into the future have been computed so far – even though much more context is already available. That is, we generate structure *just-in-time*, only shortly before it is needed by the next processor. This holds very similarly for the vocoding step that produces the speech signal just-in-time.

The just-in-time processing approach, combined with the increasing temporal granularity of units towards the lower levels has several advantages: (a) little utterance-initial processing (only what is necessary to produce the beginning of the signal) allows for very responsive systems; and (b) changes to the initial plan result only in a modest processing overhead because little structure has to be re-computed.

4.3 Technical overview

As a basis, we use MaryTTS (Schröder and Trouvain, 2003), but replace Mary's internal data structures and processing strategies with structures from our incremental SDS architecture, the INPROTK toolkit (Schlangen et al., 2010; Baumann and Schlangen, 2012b), which implements the IU model for incremental dialogue processing (Schlangen and Skantze, 2009). The model conceptualises – and the toolkit implements – incremental processing as the processing of *incremental units* (IUs), which are the smallest 'chunks' of information at a specific level (the boxes in Figure 2). IUs are interconnected to form a *network* (e. g., words keep links to their associated phonemes and neighbouring words and vice-versa) which represents the system's information state.

The component is fed with chunk IUs which contain some words to be synthesised (on their own or appended to an ongoing utterance). For simplicity, all units below the chunk level are currently generated using Mary's (non-incremental) linguistic preprocessing capabilities to obtain the target phoneme sequence. For continuations, the preceding parts of the utterance are taken into account when generating prosodic characteristics for the new chunk. Also, our component is able to revoke and exchange chunks (or unspoken parts thereof) to change what is to be spoken; this capability however is not used in the example system presented in Section 5.

The lowest level module of our component is what may be called a *crawling vocoder*, which actively moves along the phoneme IU layer and executes two processing steps: (a) for each phoneme it generates the sequence of HMM parameter frames using a local optimisation technique (using up to four neighbouring phonemes as context) similar to the one described by Dutoit et al. (2011); and (b) vocoding the HMM parameters into an audio stream which contains the actual speech signal.

IUs have a 'progress' field which is set by the crawling vocoder to one of 'upcoming', 'ongoing', or 'completed', as applicable. IUs provide a generic update mechanism to support notification about progress changes in delivery. The next section describes how this is used to drive the system.

5 Integrating iNLG and iSS for Adaptive Information Presentation

Integrating incremental microplanning with incremental speech synthesis in one incremental output generation architecture allows us to test and explore how their capabilities act in a coordinated way. As a first example, we implemented a system that presents information about events in an appointment database (e. g., new, conflicting or rescheduled appointments) and is able to cope with external noise burst events, as they might for example occur on a bad telephone line or when using a dialogue system next to a busy street. The focus is on the incremental capabilities of the system which enable its adaptive behaviour.

5.1 Component interplay

iNLG and iSS are implemented as IU modules in the INPROTK architecture. The control flow of the system (Figure 3) is managed without special coupling between the modules, relying only on the left-to-right processing capabilities of INPROTK combined with its generic IU update mechanism for transporting feedback from iSS to iNLG. Both modules can be (and have been) combined with other IU modules.

To communicate an appointment event, the iNLG module starts by generating two initial chunk IUs, the first to be expressed immediately, the second as additional prosodic context (chunk lengths differ with an average of about 4 words). The iNLG registers as a 'progress listener' on each chunkIU, which registers

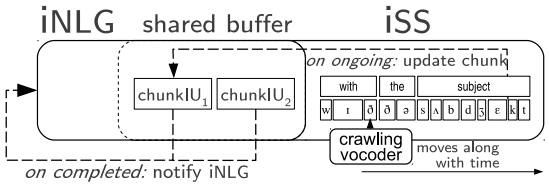

Figure 3: Information flow (dashed lines) between iNLG and iSS components (rounded boxes) and incremental units (rectangular boxes). The vocoder crawls along with time and triggers the updates.

Table 2: Processing time per processing step before delivery can begin (in ms; averaged over nine stimuli taking the median of three runs for each stimulus; calculated from log messages; code paths preheated for optimisation).

	non-incr.	**incr.**
NLG-microplanning	361	52
Synthesis (ling. pre-processing)	217	447[2]
Synthesis (HMM and vocoding)	1004	21
total response time	1582	519

as a progress listener on a phonemeIUnear its end. Shortly before iSS finishes speaking the chunk, iNLG is thus informed and can generate and send the next chunk to iSS *just-in-time*.

If adaptation to noise is needed, iNLG re-generates and re-sends the previous chunk, taking altered parameters into account. Again, a subsequent chunk is immediately pre-generated for additional prosodic context. This way of generating sub-utterance chunks ensures that there is always one chunk lookahead to allow the iSS module to compute an adequate intonation for the current chunk, while maintaining the single chunk as increment size for the system and minimising redundant work on the side of iNLG (this lookahead is not required for iSS; but if it is unavailable, sub-utterance chunks may be inadequately connected prosodically).

5.2 Responding to a noise event

A third module, the noise detector connects to both iSS and iNLG. On noise onset, it informs iSS to interrupt the ongoing utterance after the current word (this works by breaking the links between words so that the crawling vocoder finishes after the currently ongoing word). Once a noise burst ends, iNLG is informed, re-generates the interrupted sub-utterance chunk with the verbosity level decreased by one and the assumed understanding value increased by one (this degree of adaptation results in a noticeable difference, it is, however, not based on empirical study). The values are then reset, the following chunk is generated and both chunks are sent to iSS.

It should be noted, that we have not implemented a real noise source and noise detector. Instead, our random noise simulator generates bursts of noise of 1000 ms after a random time interval (between 2 and 5 seconds) and directly informs the system 300 ms after noise starts and ends. We think it is reasonable to assume that a real noise detector should be able to give accurate information with a similar delay.

6 Evaluation

6.1 Quantitative evaluation

One important argument in favour of incremental processing is the possibility of speeding up system response time, which for non-incremental systems is the sum of the times taken by all processors to do their work. An incremental system, in contrast, can *fold* large amounts of its processing time into the ongoing speech output; what matters is the sum of the *onset times* of each processor, i. e., the time until a first output becomes available to the next processor.

Table 2 summarises the runtime for the three major steps in output production of our system using nine utterances from our domain. Both NLG and speech synthesis' onset times are greatly reduced in the incremental system.[2] Combined, they reduce system response time by more than a second. This is mostly due to the almost complete folding of HMM optimisation and vocoding times into the spoken utterance. NLG profits from the fact that at the beginning of an utterance only two chunks have to be generated (instead of an average of 6.5 chunks in the non-incremental system) and that the first chunk is often very simple.

6.2 Subjective evaluation

To further test whether the system's behaviour in noisy situations resembles that of a human speaker

[2]The iSS component by mistake takes the symbolic preprocessing step twice. Unfortunately, we found this bug only after creating the stimuli used in the subjective evaluation.

in a similar situation, we let humans rate utterances produced by the fully incremental, adaptive system and utterances produced by two non-incremental and less responsive variants (we have not used non-incremental TTS in combination with iNLG as another possible base-line as pretests showed this to sound very unnatural due to the missing prosodic linkage between phrases). The participants were to rate whether they agree to the statement *'I found the behaviour of the system in this situation as I would expect it from a human speaker'* on a 7-point Likert-scale.

In condition A, full utterances were generated non-incrementally, synthesised non-incrementally and played without responding to noise-interruptions in the channel (as if the system did not notice them). Utterances in condition B were generated and synthesised as in condition A, but playback responded to the noisy channel, stopping when the noise was noticed and continuing when noise ended. For condition C, utterances were generated with the fully incremental and adaptive system described in Section 5. Upon noise detection, speech synthesis is interrupted and, when the noise ends, iNLG will re-generate the interrupted sub-utterance chunk – using the adaptation strategy outlined in Section 5.2. This then triggers iSS into action and shortly after, the system continues speaking. Nine system runs, each producing a different utterance from the calendar domain, were recorded in each of the three conditions, resulting in a total of 27 stimuli.

Before the actual stimuli were presented, participants listened to two example stimuli without noise interruptions to get an impression of how an average utterance produced by the system sounds. After the presentation of these two examples, the 27 stimuli were presented in the same random order. Participants listened once to each stimulus and rated it immediately after every presentation.

Twelve PhD-students (3 female, 9 male; mean age 30.5 years; 11 with German as one of their first languages; none with uncorrected hearing impairment) from Bielefeld University participated in our study and listened to and rated the 27 stimuli.

A Friedman rank sum test revealed a highly significant difference between the perceived human-likeness of the three systems ($\chi^2 = 151, p < .0001$). Median values of stimulus ratings in the conditions *A*, *B* and *C* were 2, 2 and 6 respectively, indicating that the fully incremental system was rated considerably more human-like. This was also shown through a *post-hoc* analysis with Wilcoxon signed rank tests which found no significant difference between condition A and B ($V = 1191.5, p = .91$)[3]. Conditions A and C, however, differed highly significantly ($V = 82, p < .0001$), as did conditions B and C ($V = 22.5, p < .0001$) – even after applying a Bonferroni correction to correct for a possible cumulation of α-errors.

7 Conclusion

We have presented what is – to the best of our knowledge – the first integrated component for incremental NLG and speech synthesis and demonstrated the flexibility that an incremental approach to output generation for speech systems offers by implementing a system that can repair understanding problems.

From the evaluation we can conclude that incremental output generation (both iNLG and iSS in isolation or combined) is able to greatly speed up system response time and can be used as a means to speed up system response even in an otherwise non-incremental system. Furthermore, we showed that the behaviour of our fully incremental and adaptive system was perceived as significantly more human-like than the non-incremental and the non-incremental but responsive baseline systems.

The understanding problem that our demonstrator system tackled was of the simplest kind, namely acoustic non-understanding, objectively detectable as the presence of noise. In principle, however, the same mechanisms of stopping and rephrasing can be used to tackle more subjective understanding problems as can be signalled by linguistic feedback. Our incremental output generation component gives us an ideal basis to explore such problems in future work.

Acknowledgements This research is partially supported by the Deutsche Forschungsgemeinschaft (DFG) in the Center of Excellence in 'Cognitive Interaction Technology' (CITEC) and through an Emmy Noether Fellowship to the last author.

[3]This suggests that it does not matter whether a system responds to problems in the communication channel by waiting or totally ignores these problems. Notice, however, that we did not test recall of the calendar events. In that case, condition B should outperform A, as some information was clearly inaudible in A.

References

Jens Allwood, Joakim Nivre, and Elisabeth Ahlsén. 1992. On the semantics and pragmatics of linguistic feedback. *Journal of Semantics*, 9:1–26.

Rachel Baker, Alastair Gill, and Justine Cassell. 2008. Reactive redundancy and listener comprehension in direction-giving. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 37–45, Columbus, OH.

Timo Baumann and David Schlangen. 2012a. INPRO_iSS: A component for just-in-time incremental speech synthesis. In *Proceedings of ACL System Demonstrations*, Jeju, South Korea.

Timo Baumann and David Schlangen. 2012b. The INPROTK 2012 release. In *Proceedings of the NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data*, pages 29–32, Montréal, Canada.

Hendrik Buschmeier and Stefan Kopp. 2011. Towards conversational agents that attend to and adapt to communicative user feedback. In *Proceedings of the 11th International Conference on Intelligent Virtual Agents*, pages 169–182, Reykjavik, Iceland.

David DeVault. 2008. *Contribution Tracking: Participating in Task-oriented Dialogue Under Uncertainty*. Ph.D. thesis, Rutgers, The State University of New Jersey, New Brunswick, NJ.

Thierry Dutoit, Maria Astrinaki, Onur Babacan, Nicolas d'Alessandro, and Benjamin Picart. 2011. pHTS for Max/MSP: A streaming architecture for statistical parametric speech synthesis. Technical Report 1, numediart Research Program on Digital Art Technologies, Mons, Belgium.

Jens Edlund. 2008. Incremental speech synthesis. In *Second Swedish Language Technology Conference*, pages 53–54, Stockholm, Sweden, November. System Demonstration.

Markus Guhe. 2007. *Incremental Conceptualization for Language Production*. Lawrence Erlbaum, Mahwah, NJ.

Christine Howes, Matthew Purver, Patrick G. T. Healey, Gregory Mills, and Eleni Gregoromichelaki. 2011. On incrementality in dialogue: Evidence from compound contributions. *Discourse & Dialogue*, 2:279–311.

Gerard Kempen and Edward Hoenkamp. 1987. An incremental procedural grammar for sentence formulation. *Cognitive Science*, 11:201–258.

Anne Kilger and Wolfgang Finkler. 1995. Incremental generation for real-time applications. Technical Report RR-95-11, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.

Willem J. M. Levelt. 1989. *Speaking: From Intention to Articulation*. The MIT Press, Cambridge, UK.

Kyoko Matsuyama, Kazunori Komatani, Ryu Takeda, Toru Takahashi, Tetsuya Ogata, and Hiroshi G. Okuno. 2010. Analyzing user utterances in barge-in-able spoken dialogue system for improving identification accuracy. In *Proceedings of INTERSPEECH 2010*, pages 3050–3053, Makuhari, Japan.

Ehud Reiter and Somayajulu Sripada. 2002. Human variation and lexical choice. *Computational Linguistics*, 28:545–553.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–718, Athens, Greece.

David Schlangen, Timo Baumann, Hendrik Buschmeier, Okko Buß, Stefan Kopp, Gabriel Skantze, and Ramin Yaghoubzadeh. 2010. Middleware for incremental processing in conversational agents. In *Proceedings of SIGdial 2010: the 11th Annual Meeting of the Special Interest Group in Discourse and Dialogue*, pages 51–54, Tokyo, Japan.

Marc Schröder and Jürgen Trouvain. 2003. The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*, 6:365–377.

Gabriel Skantze and Anna Hjalmarsson. 2010. Towards incremental speech generation in dialogue systems. In *Proceedings of SIGDIAL 2010: the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–8, Tokyo, Japan.

Matthew Stone, Christine Doran, Bonnie Webber, Tonia Bleam, and Martha Palmer. 2003. Microplanning with communicative intentions: The SPUD system. *Computational Intelligence*, 19:311–381.

Matthew Stone. 2002. Lexicalized grammar 101. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 77–84, Philadelphia, PA.

Paul Taylor. 2009. *Text-to-Speech Synthesis*. Cambridge Univ Press, Cambridge, UK.

Tomoki Toda and Keiichi Tokuda. 2007. A speech parameter generation algorithm considering global variance for HMM-based speech synthesis. *IEICE TRANSACTIONS on Information and Systems*, 90:816–824.

Keiichi Tokuda, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. 2000. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proceedings of ICASSP 2000*, pages 1315–1318, Istanbul, Turkey.

Marylin Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.