In: *Journal on Multimodal User Interfaces*, 2014, DOI 10.1007/s12193-013-0130-3
The final publication is available at link.springer.com

An Architecture for Fluid Real-time Conversational Agents: Integrating Incremental Output Generation and Input Processing

Stefan Kopp · Herwin van Welbergen · Ramin Yaghoubzadeh · Hendrik Buschmeier

Abstract Embodied conversational agents still do not achieve the fluidity and smoothness of natural conversational interaction. One main reason is that current system often respond with big latencies and in inflexible ways. We argue that to overcome these problems, real-time conversational agents need to be based on an underlying architecture that provides two essential features for fast and fluent behavior adaptation: A close bi-directional coordination between input processing and output generation, and incrementality of processing at both stages. We propose an architectural framework for conversational agents (ASAP) providing these two ingredients for fluid real-time conversation. The overall architectural concept is described, along with specific means of specifying incremental behavior in BML and technical implementations of different modules. We show how phenomena of fluid real-time conversation, like adapting to user feedback or smooth turn-keeping, can be realized with ASAP and we describe in detail an example real-time interaction with the implemented system.

S. Kopp · H. van Welbergen · R. Yaghoubzadeh · H. Buschmeier
Sociable Agents Group, CITEC and Faculty of Technology, Bielefeld University
PO-Box: 10 01 31, 33501 Bielefeld, Germany
E-mail: skopp@techfak.uni-bielefeld.de
ORCID: http://orcid.org/0000-0002-4047-9277

H. van Welbergen
E-mail: hvanwelbergen@techfak.uni-bielefeld.de
ORCID: http://orcid.org/0000-0002-4122-062X

R. Yaghoubzadeh
E-mail: ryaghoubzadeh@uni-bielefeld.de
ORCID: http://orcid.org/0000-0002-3918-6725

H. Buschmeier
E-mail: hbuschme@uni-bielefeld.de
ORCID: http://orcid.org/0000-0002-9613-5713

Keywords Embodied Conversational Agents Architecture · Fluid Real-time Interaction · Generation–Interpretation Coordination · Incremental Processing · ASAP · BMLA

1 Introduction

Embodied Conversational Agents (ECAs) aim to enable human-like face-to-face interaction between a human user and an artificial agent. Cassell and colleagues [6] define them to have abilities for (1) "recognizing and responding to verbal and nonverbal input," (2) "generating verbal and nonverbal output," (3) "dealing with conversational functions such as turn taking, feedback, and repair mechanisms," and (4) contributing "signals that indicate the state of the conversation or contribute new propositions to the discourse". A large body of work has been directed to developing approaches to these individual challenges. Still, interacting with today's ECAs is for the most part characterized by noticeable latencies and slow response times, resulting in an unnatural clumsiness. In other words, ECAs often fall short of the fluidity and smoothness of natural human conversational interaction.

One cornerstone of human conversation is the high interactivity and real-time responsiveness with which cooperative interactants co-construct their contributions. For example, while producing communicative actions, speakers attend to and even elicit reactions from the addressee. Depending on this instant feedback, speakers can re-plan the remaining part(s) of their communicative plan, adapt it to the addressees' needs, put it on hold, insert a sub-dialogue, and continue at the point of interruption. All this is done in such an effortless, smoothly coordinated and seemingly natural way that it is not even apparent that difficulties were payed attention to or that plans were changed mid way. That is, acting in a conversation can only partially be based on extensive planning ahead and deep representational models. Instead, interaction partners, while being guided by overall goals and strategies, are also

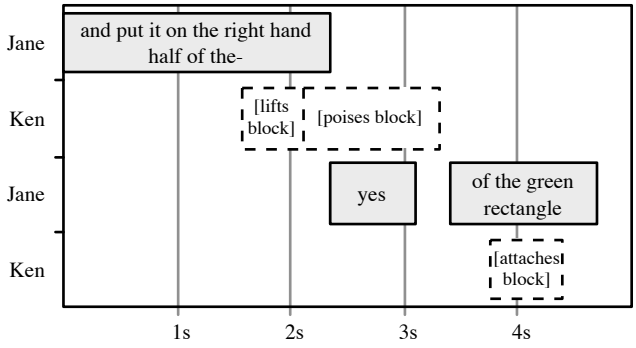

Fig. 1 Example of a speaker monitoring the addressee's feedback and adapting her utterance (redrawn from [7, p. 74]).

highly sensitive to the the partner's verbal and nonverbal feedback and are able to alter their utterances in progress accordingly [7]. Fig. 1 shows an example from this study, in which Jane (the speaker) suspends her utterance in response to Ken's (the addressee) feedback, inserts a short feedback utterance herself, and then resumes her initial delivery. These abilities are crucial for human-like, real-time interactivity and conversational fluidity, but they exceed the abilities of current ECAs.

We argue that, to achieve these abilities in ECAs, what is needed are two pivotal features at the architectural level: (1) incremental processing and (2) bi-directional, multi-level interaction between input processing and output generation. In this paper, we discuss in detail the requirements that imply such an architecture (Sect. 2), discuss related work (Sect. 3), and propose the ASAP (*Artificial Social Agent Platform*) architectural framework (Sect. 4). ASAP is as a dedicated ECA middleware specifically tailored to real-time conversation. It lays down at a technical but also a semantic-conceptual level how the input and output processing modules need to interact with each other. It thus extends the BML/FML output generation framework [23,48] and goes beyond, on the one hand, technical software architectures for multi-component ECA systems (as the Virtual Human Toolkit [15]) and, on the other hand, standard multi-agent communication architectures (e.g., blackboard systems). We will demonstrate how ASAP enables the fluid production and adaptation of multimodal output under the presence or absence of different kinds of input from the addressee or the environment, respectively. Section 5 presents a prototype implementation of a conversational agent based on ASAP and explicates how it allows for achieving more fluid real-time interaction.

2 Architecture Requirements

Being able to have a conversational interaction with a human raises challenging demands from an artificial system. Many have noted the most obvious requirements, namely, sophisticated abilities for recognition of verbal and nonverbal input, generation of convincing multimodal output, and dealing with behavior in terms of its many conversational functions and based on an understanding of dialogue state. To cope with these demands, Cassell and colleagues [6] derive a number of architectural constraints that generalize from the multilayer, multimodal Ymir architecture [45]. The most relevant ones here are that an architecture should support receiving and transmitting multimodal information, should allow to track different threads of communication running at different timescales, and should be symmetric.

While we agree with these constraints, there are other less obvious, but nonetheless important requirements. They arise from the fact that fluid conversation hinges upon fast adaptation and coordination taking place between interlocutors. It is well known that interlocutors coordinate their behavior at very fine grained levels to collaboratively manage their interaction. For example, listeners signal their current degree of understanding or agreement, and speakers monitor this feedback and adapt their utterance flexibly and rapidly to the listener's needs. Adaptations range from different lexical choice, direct references, elaborations, or slower speech rate, to interruptions and continuation with a completely new utterance [7]. Speakers actively monitor for information about their addressee's state, and they fluently added cues to elicit such information from the listener [12].

Coordination becomes particularly visible at the moments of floor changes [33]. Turn-overlap situations are assessed on the fly and either the turn is yielded or grabbed/kept "forcibly", e.g., by adjusting the volume of speaking or the speaking rate to be sufficiently interruptive or non-interruptible, respectively [35]. In addition, interlocutors are also found to entrain (synchronize) their behavior timing or align their behaviors with regard to form and meaning (cf. [21]). Reported examples of this are the convergence of speech rate or response latency [43], and the alignment in lexical choice or of grammatical forms [11].

These fast adaptations rest on speakers producing their speech incrementally [19]. Often, utterances are started with appositional beginnings ("uh", "well", ...), e.g., to secure the turn while not having planned out already the full contribution [33]. Likewise, listeners process an incoming utterance incrementally based on predictions about likely continuations and the speaker's intended meaning [44]. We argue that if we want to build ECAs that can engage in these fast and real-time coordinations, they need to be based on dedicated design principles as discussed in the following.

2.1 Incremental processing

Input processing must yield first results with minimal latency. To this end, it must run incrementally and form hypotheses from the so-far-perceived input. Hypotheses must be passed

on for other modules to work on them. Competing hypotheses along with their respective probabilities must be managed and continuously updated. Likewise, output generation must run incrementally, implying that incremental units are not finalized until necessary. Real-time and fluidity requirements will demand that responses be pre-planned and initiated once start conditions are met[1]. However, plans of not yet executed incremental units must be kept flexible and adaptable to minimize time-consuming re-planning. Output plans that are only partially realized (but not discarded) must be stored and be re-usable pending adaptations to ensure coherence with the unfolding discourse (as in Fig.1).

2.2 Top-down and bottom-up interaction

Incremental processing need to be combined with bi-directional information flow. Partial (bottom-up) interpretation hypotheses must be combined with the formulation and evaluation of (top-down) predictions about the interlocutor's possible next contributions. In output generation, higher-level processes must work on incremental units at larger timescales than lower-level processes. Control signals from running incremental units must be passed top-down immediately and continuously, while status updates need to be sent back bottom-up.

2.3 Linking input processing and output generation

Classically treated separately, input processing and output generation must run concurrently but with bi-directional information transfer between them. Generation of output must be not only planned top-down but must also have access to low-level information from input processing, e.g., to connect a running behavior with specific perceptual information (as classically done via a reactive route to enable gaze-tracking a moving target [45]). For example, if own actions are currently running or are already prepared, attention must be focused on those input features that need to be assessed as to whether (pre-)conditions for these actions are being fulfilled or violated. Likewise, input processing must be informed about concurrent behavior generation, e.g., to be able to focus on subtle cues that would normally be not salient or hidden in noise, to tune in to expected input patterns or those patterns that have been used previously by the output production systems, to compensate for self-induced sensory noise, or to provide additional context that can help to resolve ambiguities between parallel, competing hypotheses.

[1] Technically it may also be advantageous to prepare possible alternative responses in parallel.

3 Related Work

A large number of ECAs have been developed based on architectures that in some way integrate input processing, output planning, and output generation. The architecture proposed by Cassell and colleagues [6] consisted of an input manager that fuses audio, speech and gestural inputs and sends them to a deliberative module in charge of creating behavior based on proper input interpretation and output planning. In addition, a reaction module realizes a direct pathway from input to output processing to implement hardwired quick reactions. Processing along both pathways operates upon self-contained plans or specifications of whole input or output behavior sets. For example, output generation produces a schedule of behaviors, either in absolute-time based or event-based ways, that specifies in advance how a number of extended behavior are to unfold over time.

Almost all existing ECAs have followed this general architecture layout, which is characterized by a modular structure and non-incremental, strictly sequential processing along a deliberative or a reactive route (e.g., Max [22], Greta [16], Virtual Justina [20]). Very recent systems still adopt this structure: In the Thalamus architecture [32], perception is simply a unidirectional information link from the body interface to the mind interface; in de Kok and Heylen's feedback-giving system [9] input arrives in a module which determines the appropriateness of a listener response; in Crook et al.'s 'How Was Your Day?' prototype [8] for coping with barge-ins, a 'long' loop is used for intent planning and a shorter loop is used to handle interruptions, back-channel feedback and emotional mirroring. Like others, this agent does not process input or generate output incrementally and the authors note that the use of incrementality would have made their design more elegant.

Some first attempts to overcome this rigid style of processing have been made. Nijholt et al. [28] proposed a first approach to directly link the timing of ongoing behavior to the timing *predictions* of interlocutor events. This enabled a finer degree of temporal coordination with the user's motion, as demonstrated in a dancer agent, a virtual orchestra conductor and a virtual fitness trainer. Incremental processing is also increasingly assumed to be a key principle for natural dialogue modeling. Schlangen and Skantze [38] propose that the internal mechanisms in incremental dialogue agents can be described in terms of abstract modules that communicate by passing around and holding *incremental units* (IUs). Several implementations of this model have been developed [37] and many aspects of dialogue agents have been successfully modeled within this incremental processing framework (speech recognition, natural language understanding [1], dialogue management [5,46], natural language generation [41, 3], speech synthesis [2]). However, implementations of the IU

architecture have been uni-modal so far (typically speech/text only).

Incremental realization of multimodal behavior by automatically connecting chunks that retain their internal synchrony, was first proposed in the ACE system [24]. The realizer component in our ASAP architecture [47] extends this approach by providing more fine-grained behavioral units that can be linked to anticipated input events, following [28].

Much effort has been put on standardizing the output generation of ECAs in a way that allows for architectural modularity and flexibility of generated behavior. The SAIBA initiative [23] has put forth three main stages (content planning, behavior planning, behavior realization) with standardized XML interface representations in-between (BML, FML). This has now become the de facto standard for behavior generation in ECAs. However, this model is deliberately left coarse and does not make any assumptions about input/output coordination nor incremental processing. Recent work has started to extend this framework to also include a representation of nonverbal input behavior (the Perception Markup Language, PML) [36]. In contrast to BML, PML handles multiple hypothesis on three levels of input processing (sensing, behaviors and functions) along with probabilities to represent uncertainties. In a first example architecture, the different levels of PML are used to inform dialogue state update, intent planning or behavior generation.

The question how different stages of input processing and output generating should ideally interact within conversational agents has received only little attention so far. In principle, this includes horizontal (between input/output modules) as well as vertical (bottom-up and top-down) interactions. It is well known that in humans perception influences action via direct links that can bypass cognition, that perception and action share common representations, or that action planning influences perception (see e.g. [49]). Cognitive robotics has adopted this view more extensively than research on ECAs, while focusing on the control of manipulative or locomotive actions [18]. Haazebroek, van Dantzig and Hommel [14] presented a computational model which uses shared representations between perception and action for cognitive robots. Sadeghipour and Kopp [34] have presented a probabilistic model for learning, recognizing and producing communicative gestures based on shared sensorimotor representations. Hoffman and Breazeal [17] implement task anticipation for collaborative robotic systems and use knowledge about past events as a top-down bias in perception, allowing for more accurate and rapid 'sensing' decisions. This is recently paralleled in the field of human language technology (speech recognition or language understanding) which has started to build dynamic language models (vocabularies, grammars, recognition weights) in order to increase accuracy and adaptivity [39]. Such models are adjusted at runtime, e.g., when dialogue context changes [40], certain entities become salient [26], or conversation threads are switched [25].

In sum, promising approaches are increasingly put forward for incremental language processing as well as for more flexible perception-action integration. However, to the best of our knowledge, a comprehensive concept of how to combine these methods in an ECA architecture that enables fast responses and incremental adaptation as needed for fluid, real-time multimodal conversation is still lacking.

4 The ASAP Framework

We are developing a framework for embodied agents that are to engage in human-like fluid conversation, with a degree of real-time flexibility and smooth interactivity similar to what humans are used to and expect from each other—the *Artificial Social Agent Platform* (ASAP). We want to have a principled architectural basis that provides the necessary degree of incremental processing as well as integration of output generation and input processing in a conceptually well-defined and technically sound way. ASAP can be seen as an extension of the SAIBA framework to a full input-output architecture with incremental processing. This includes an extension of the representational models (most notably, BML) to the specific problems of incremental specification and generation; for the description of perception events we aim to make use of a standardized representation language like the recently initiated Perception Markup Language (PML) [36]. In the remainder of this section, the overall architecture as well as our approach to incremental behavior specification and processing are explained.

4.1 Integrated architecture

The architectural requirements we have pointed out above require incremental processing in a *strongly linked* architectural layout. That is, modules are connected with bi-directional links, both "horizontally" between input processing and output generation as well as "vertically" between different stages of processing. The ASAP architecture (see Fig. 2) is designed as such an architecture. This enables input processing to tune in on specific output patterns (priming) or to be sensitized for specific time windows. The use of bi-directional, multi-layer links also allows us to cover the SAIBA framework with our architecture and thus to profit from the reusability it affords.

The overall architecture layout is shown in Fig. 2. The left-hand side comprises the behavior generation sub-system, the right-hand side the behavior processing sub-system. All modules are supposed to run concurrently and to communicate by means of asynchronous, incremental messages as described in the next section. During generation the Intent Planner specifies communicative goals, intended messages

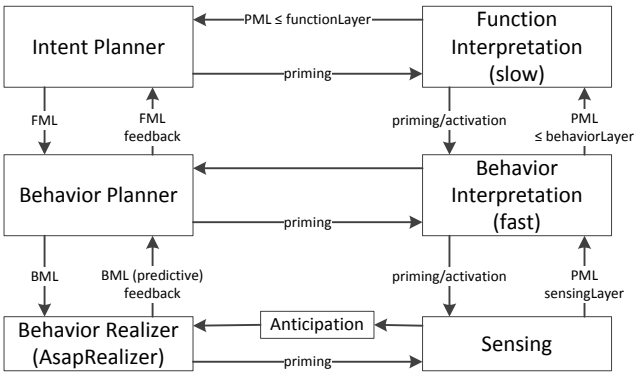

Fig. 2 Outline of the ASAP integrated architecture.

and interactional goals in FML and is in charge of keeping track of the discourse state including the grounding status of propositions. A Behavior Planner translates intentions into surface behaviors specified in BML. The Behavior Realizer takes BML behavior specifications and transforms them into overt behavior of an ECA.

As the behavior is planned and executed by the Realizer, predictive feedback on the current state of the ongoing behaviors is sent back to the Behavior Planner. Such information is essential for continuing after an interruption (e.g., to not repeat what has already been said) or deciding whether to yield the turn (e.g., if the essential information has been delivered) or use some strategies to keep it. The Behavior Planner provides the Intent Planner with FML feedback to inform it of the ongoing progress in achieving the requested intentions.

The Sensing Module receives sensory data and turns them into descriptions at the PML sensing layer [36]. This steps involves, e.g., speech recognition or the extraction of significant features of a gesture. The results are processed first by (fast) Behavior Interpretation processes and subsequently by (slower) Function Interpretation module. Fast interpretation map distinct and unique behavioral patterns (e.g., nodding, blushing) onto fixed interpretations. Such observed behaviors directly inform the Behavior Planner in its selection of new behavior or the modification of ongoing output behavior. For example, the Behavior Planner may use this information to instruct the Realizer to stop speaking when receiving a long (but otherwise unanalyzed) speech fragment from the user. PML behaviors are sent to the Function Interpretation component, which integrates them into higher-level functions (e.g., agreement, shame). These functions inform the Intent Planner.

The architecture explicitly enables top-down information flow in the input processing subsystem: The Function Interpretation module can prime the observation of certain behaviors in the Behavior Interpretation module, based on the functional state the user is observed or predicted to be in. For example, if the user is listening, attending to typical listening behaviors (e.g., nodding, saying "uh huh") may be primed. To this end, the Behavior Interpretation module can activate or adjust sensing processes. This accommodates dynamic language models in speech recognition, which can be activated to specifically facilitate the early and robust recognition of missing parts of different hypothesized user contributions (as in [25]).[2]

Furthermore, the ASAP architecture explicitly allows information to flow from output generation to input processing and back, at all stages. Input processing modules can thus profit from "priming" through the generation modules. For example, the Realizer may prime the recognition of certain words in a keyword spotter or speech recognizer in the Sensing Module if these words have previously been used by the ECA itself (if coordination occurs between speaker and listener, a human listener will likely adopt the terminology used by the virtual speaker [11]). The Behavior Planner can prime the recognition of feedback behaviors (e.g., nodding, saying "uh huh") that are expected to complement an adjacency pair opened by the agent, or whenever the agent decides to execute feedback eliciting behavior. Finally, the Intent Planner may prime the interpretation of behavior in terms of task-related functions (as in [17]).

4.2 Incremental generation and processing middleware

A pivotal feature of ASAP is its inherently incremental output generation and input processing. This ability rests upon a middleware (IPAACA) for exchanging linked *incremental units* (IUs) between processing components, following the general abstract model proposed for incremental dialogue processing [38]. IUs can be established and categorized by any module, and are immediately available to all modules that have registered for their category. Once established, IUs persist as a bi-directional communication link until one of the modules retracts them. Modules fill IUs incrementally with their output, thus allowing the receiving modules to operate concurrently on uncompleted input. In addition, IUs can be connected using specific links. That way, lower level constructs (e.g., behaviors) are *grounded* in the top level constructs that caused and specified them (e.g., intentions). This allows for changes to the lower level constructs in a top-down fashion, using commands that act upon the top-level constructs, while at the same time propagating information from the lower-level constructs (such as the self-monitored state of execution) to their higher-level correlates. In addition to cross-level grounding links, intra-level relations between constructs can be captured using other links, such as *dependence* on the intent planning level or, analogously, *sequence* on the behavior planning level.

[2] Such activations can be realized via priming by the Intent Planner and Function Interpretation.

Since incremental input processing has received a significant amount of attention in the spoken dialogue systems community [38, 1], and we have successfully started to integrate this work into ASAP (see Sect. 5), we focus on how IPAACA can be used to model and control the incremental generation of multimodal output across the levels of the ASAP architecture: Intentions, behavior plans, and motor control units of realized behaviors are modeled as IUs residing at the corresponding level. Top-down information flow is realized asynchronously, by registering specifications, modifications and retractions (of intentions, plans, behaviors) by top-level modules. Bottom-up information is shaped by dedicated update mechanisms. Generation progress on the realization layer, monitored on a fine-grained timescale, is mapped to status information for behaviors at the behavior planning level. These states are mapped onto success information at the intent level, while checking the constraints set forth by same-level links, e.g., that a conversational intention is fulfilled with maximum probability when all the behaviors originally planned as a sequence have been reported completed.

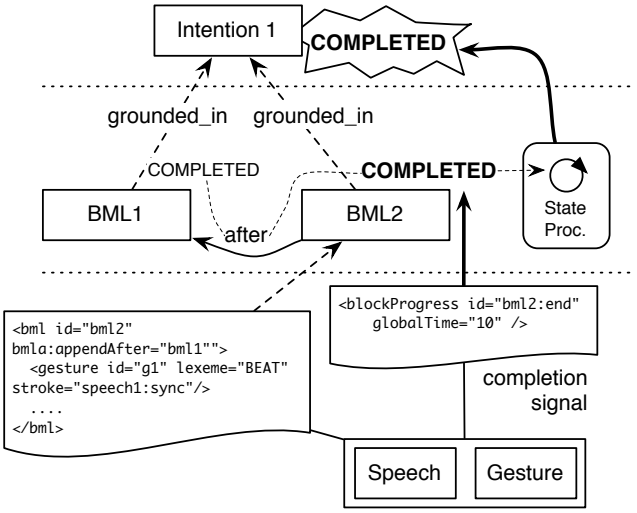

Fig. 3 Updating of multi-level incremental units of generated output.

4.3 Behavior Specification

BML messages and BML feedback messages are used as incremental interface representation between the levels of output generation. However, several extensions to BML are needed to allow for fluent, on-the-fly changes to ongoing behavior. We have defined the following extension BMLA, which subsumes regular BML and is realized in the current implementation of our ASAP realizer [47].

BMLA messages can be used to either construct a new increment (possibly as one of several execution alternatives), to (gracefully) interrupt an ongoing increment, to activate a previously pre-planned increment, or to partly adapt an ongoing increment (e.g., speak louder). BMLA feedback messages inform the Behavior Planner of the predicted timing and shape of planned increments and (once they are executed) of the timing and content of their actual realization. Execution failures are also communicated through these bottom-up messages. To allow perception to directly inform behavior generation, BMLA further allows a direct link between the timing of a behavior and anticipated timing of events sensed by the Sensing Module.

Table 1 provides an overview of the proposed extensions (see the BMLA Wiki[3] for a more extensive treatment of their syntax and semantics). Since BML was originally designed for self-contained specifications, which contradicts the idea of incremental composition, we choose BMLA blocks of appropriate size to be our incremental units of behavior planning (typically an intonation phrase and one or more aligned gestures and facial expressions), and we introduce additional means of specifying inter-block relations that arise in incremental generation (Table 1, row 5). Furthermore, we provide a pre-planning mechanism (Table 1, row 1) that allows us to prepare BML blocks that contain likely execution paths given current input increments. These blocks can then be instantly activated when required, enabling smooth and fluid reactions to user behavior.

As we have already seen (Fig. 1), behavior can be interrupted or modified instantaneously, and later be resumed, even in the midst of a syntactic unit. That is, modification of executed behavior may well happen with finer granularity than that provided by BML blocks. The ASAP realizer therefore allows for flexible top-down modification (Table 1, row 3) and interruption of running BMLA blocks (see Table 1, row 2, for the BMLA specification mechanisms for this; cf. [47] for an implementation of graceful gesture interruption). Moreover, we extend BML to enable direct integration between Sensing and Behavior Realization by way of synchronizing BML behavior elements to anticipated events in the interlocutor's behavior (e.g., to let the agent start speaking slightly before the anticipated end of the interlocutor's turn; see Table 1, row 4). Likewise, BMLA feedback is provided at a finer granularity, e.g., containing the (possibly predicted and later updated) timing of gesture phases and words. Such detailed feedback informs the Behavior Planner about when what information will be delivered to the user and which information has already been delivered.

To illustrate this functionality, let us consider the example in Fig. 3. Intention 1 as well as behavior blocks BML1, BML2 are processed as IUs and connected using *grounded_in* links. BML2 is to be executed after BML1 (indicated by the same-level *after* link). This *after* link is represented in the BML block by the *appendAfter* attribute. The BML2 mes-

[3] `http://asap-project.org/wiki/BMLA`

Table 1 BMLA extensions to BML for on-the-fly behavior construction and modification.

	Specification element	Example	Description

1	`preplan` bml attribute:
	`activate` behavior:
	`onStart` bml attribute:
		`<bml id="bml1" bmla:preplan="true">...</bml>`
		`<bmla:activate id="a1" target="bml1" />`
		`<bml id="bml1" bmla:onStart="bml2">...</bml>`
			Preplan BML block bml1 for potential later activation.
			Activate preplanned BML block bml1.
			Activate preplanned BML block bml2 when bml1 starts.

2	interrupt behavior:
	
	interrupt bml attribute:
		`<bmla:interrupt id="i1" target="bml1"`
		`exclude="bml1:g1"/>`
		`<bml id="bml2" bmla:interrupt="bml1">...</bml>`
			Gracefully interrupt all behaviors but g1 in bml1.
			
			Gracefully interrupt all behaviors in bml1.

3	`parametervaluechange` behavior:	`<bmla:parametervaluechange target="bml1:speech1"`
		`paramId="volume" start="bml1:speech1:s1"`
		`end="bml1:speech1:s1+1">`
		`<bmla:trajectory type="linear" targetValue="90"/>`
		`</bmla:parametervaluechange>`
			Change the volume of the bml1:speech1 from its current value to 90, over a linear trajectory.

4	alignment to predicted events using `anticipators`:	`<speech id="speech1" start=`
		`"anticipators:speechStopAnt:stop-0.1">...`
		`</speech>`
			Eagerly start speaking 0.1 seconds before the predicted end of the turn of the interlocutor.

5	`appendAfter` bml attr:
	`chunkAfter` bml attr:
	`prependBefore` bml attr:
	`chunkBefore` bml attr:
		`<bml id="bml2" bmla:appendAfter="bml1">...</bml>`
		`<bml id="bml2" bmla:chunkAfter="bml1">...</bml>`
		`<bml id="bml1" bmla:prependBefore="bml2">...</bml>`
		`<bml id="bml1" bmla:chunkBefore="bml2">...</bml>`
			Append bml2 after bml1, no co-articulation.
			'Chunk' bml2 after bml1, co-articulation allowed.
			Prepend bml1 before bml2, no co-articulation.
			'Chunk' bml1 before bml2, co-articulation allowed.

sage (Fig. 3, l.h.s.) is used to construct a plan in the Realizer that contains the motor control units (including speech and gesture) that compose BML2. These motor control units are again grounded in BML2, so that new BML messages can refer to them (e.g., to adapt the motor plan or interrupt elements of it). The Realizer uses the grounding information to provide feedback messages to inform the Behavior Planner that the execution of BML2 is completed. Since the Behavior Planner knows (by means of the *grounded_in* links) that all IUs required to satisfy Intention 1 are completed, it can now inform the Intent Planner that this intention is achieved (using FML feedback).

The IU information structure also allows a direct response to arriving or even missing feedback from the addressee (via links from the input processing modules), as we have seen in the early example in Fig. 1. For example, after referring to an object in the environment, the recognition of a lack of acceptance by the addressee can be viewed as an obstruction to successful plan execution. In terms of our generation architecture, this is modeled as a retraction of a hypothesis node that arose from an assumed high probability of the referenced object being in the common ground. The retraction event would spread to those plans grounded in it, which in turn would notify the currently planned behaviors, which in turn would notify those in execution, eventually interrupting them. However, the interrupted plan need not be fully discarded. After the new intention (here: to make sure the object of the action is in the common ground) has been fulfilled, those plans that were interrupted can be resumed without complete replanning. Continuous estimation of such status information and mapping it onto different measures, e.g., of completeness on the behavior planning level or failure/success on the intent planning level, can only be done locally within one component, but becomes possible through the IU information structure.

The next section describes in detail how these mechanisms enable a number of phenomena of fluid, real-time conversational interaction in an implemented system.

5 Implementation and Results

The ASAP framework has been implemented in a first ECA prototype system, with different components being realized in different depth and detail. It is based on the incremental processing middleware IPAACA [37] and comprises an incremental language generation system [3], and an advanced BML Behavior Realizer [47]. The other components are operational as prototype implementations to allow evaluating the framework. Several third party (incremental) input processing and behavior generation components have been embedded in this and other prototypes: Microsoft Speech SDK used in incremental input processing mode, the Mary-based incremental TTS inpro_iSS [2], and the openSmile audio feature extractor [10]. We have incrementalized the SPUD sentence planner [42] so that—given an appropriately structured specification of a communicative intention—it generates natural language sentences in chunks of the size of intonation units; see [3] for details.

The ASAP behavior realizer drives our virtual agent BILLIE (Fig. 8) enabling a face-to-face conversation setting. Predecessors of this realizer have already been used in others tightly coordinated interactional settings, in which conversation played only a minor role, including dancing [31], conducting an orchestra [30] and performing fitness exercises together with a human user [29]. Here, using the extended version of the realizer [47] within the ASAP architecture implementation, we have realized and tested different use cases to demonstrate some of the requirements of fast, incremental behavior adaptation in fluid conversation (cf. Sect. 2).

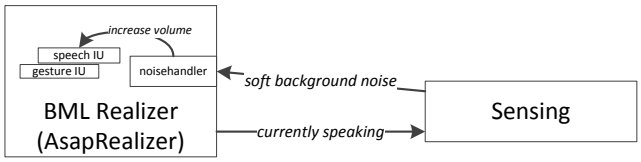

Fig. 4 Handling soft background noises.

5.1 Reacting to background noise

Suppose an ECA is to operate in a noisy environment and has to rapidly adapt to the external disturbance. In the simplest case, the agent should simply adjust its speaking volume to the noisy environment (thus implementing the Lombard effect). This can readily be accommodated by ASAP via the direct coupling between the Behavior Realizer and the Sensing Module as illustrated in Fig. 4. Here, the Realizer primes a background noise sensor in the Sensing Module, which in turn informs the Realizer of the current noise level. If a soft noise occurs, the Realizer adjusts its speaking volume accordingly.

A more interesting case is when the noise is so loud that it masks the speech even with an increased speaking volume. The agent should notice this masking, interrupt itself instantly, and then smoothly initiate a repair by repeating the masked part after the disturbance has disappeared. We have implemented this as follows (see Fig. 5 for illustration): The Realizer decides to graciously interrupt the ongoing speech and gesture (see [47] for implementation details) and to inform the Behavior Planner (through BML warning feedback) of its failure to produce (part of) the ongoing utterance in a clear enough manner. This feedback (2b), combined with knowledge of previous status updates about which parts of the utterance have already been delivered, provide the Behavior Planner with enough information to pre-plan (3a) a new BML block. This new block contains an adapted version of the interrupted intonational phrase, and may omit part of the already presented utterance (see [3] for implementation details). Fast Behavior Interpretation is then told to anticipate a noise free moment (3b) and signals Behavior Planning once it occurs (4). The adapted BML block has been pre-planned and is then activated in the BML Realizer (5). In result, the system stops speaking instantaneously when interrupted and resumes immediately with an appropriately adapted repetition as soon as the disturbance disappeared. This also shows that ASAP allows for handling external events on different levels and via different perception-action loops, depending on the needed adaptation.

5.2 Keeping the turn

Suppose that the ECA wants to keep the turn at the end of one BML block, but the user displays a (polite non-interruptive)

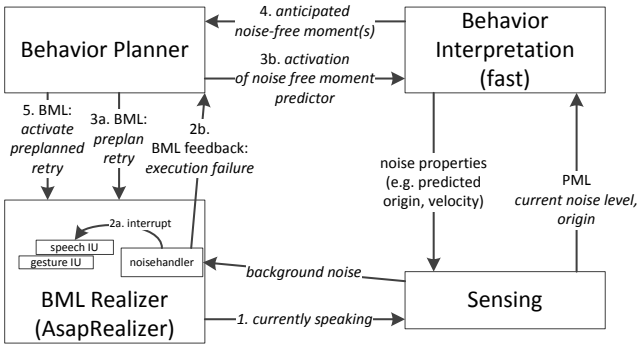

Fig. 5 Handling masking background noises.

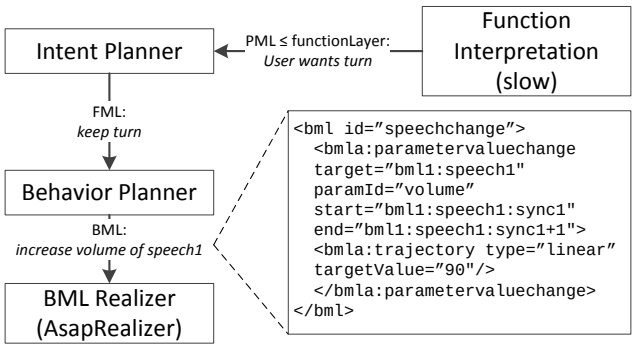

Fig. 6 Keeping the turn.

turn-wanting signal. This can be readily handled in ASAP as demonstrated in Fig. 6: the Intent Planner is informed by the Function Interpretation module that the user would like to get the turn. The Intent Planner decides to keep the turn. The Behavior Planner now continues to incrementally produce the next BML block, but with the additional adaptation of an additional `parametervaluechange` behavior, which increases the speaking volume and is synchronized with the `speech1` behavior of BML block `bml1` (here: from its starting value to target value 90 along a linear trajectory, the target value is to be reached 1 second after the custom `sync` synchronization point in `speech1`).

5.3 Adapting to user feedback

Finally, let us consider a case similar to the one discussed in Sect. 1: monitoring the user and adapting an utterance in progress to feedback from the user. This has been tackled in a previous implementation [4], but can be modeled more easily now using the ASAP framework (see Fig. 7): Here, the Intent Planner decides on a communicative goal that consists of two propositions p_1 and p_2, where p_2 presupposes p_1 to be understood (1). The Behavior Planner decides to realize both propositions in one utterance, consisting of two increments (BML blocks, intonational phrases), ip_1 and ip_2. Based on the presupposition condition, the Behavior Planner plans ip_1 and decides to append a feedback inviting cue (e.g., a pause,

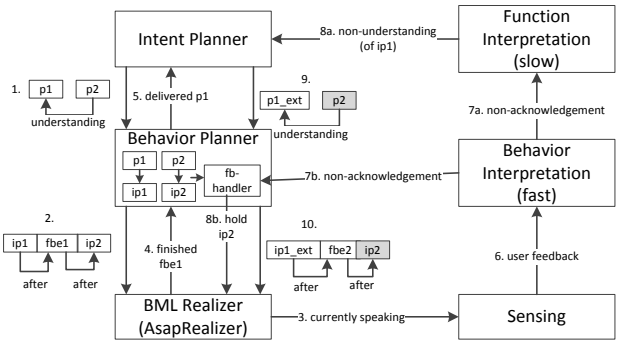

Fig. 7 Adapting to user feedback.

Fig. 8 Interacting with the prototype.

an attempt to achieve mutual gaze) to seek information on the user's understanding (2). ip_1 and ip_2 are grounded in p_1 and p_2 respectively. The Behavior Planner stores this grounding links internally. The BML Realizer starts uttering ip_1 and informs the Sensing Module that it is currently speaking, which primes feedback sensors. ip_1 and the following planned feedback inviting cue are realized and the BML Realizer notifies the Behavior Planner of this using BML feedback (4). The Behavior Planner in turn informs the Intent Planner that p_1 is delivered (but not necessarily understood or heard). Now the user provides feedback indicating problems in understanding (6). A feedback classifier in the Behavior Interpreter interprets the feedback as a non-acknowledgment (7a, 7b) and informs the Function Interpreter and the Behavior Planner.[4] Based on the grounding information of ip_1 and ip_2 and the request from the Intent Planner to only try to achieve p_2 when p_1 is understood, the feedback handler in the Behavior Planner decides to postpone the realization of ip_2 until more information on the feedback is available (8b). Such information is later provided by the Function Interpreter, which interprets the feedback as non-understanding (8a). It then starts planning an alternative for p_1 to adapt to the user's lack of understanding, for example by providing information a second time instead of an anaphoric reference (9). Finally, this adapted p_1 is transferred in a BML behavior $p1_{ext}$ and submitted to the BML Realizer (10). Note that by allowing flexible insertion of increments, no re-planning of p_2 nor ip_2 is required.

5.4 Handling interruptions in interaction

As a final result, we analyze an actual interaction with the running prototype system (see Fig. 8 for a screenshot). In this interaction, BILLIE is presenting information and is interrupted by a user's coughing. Fig. 9 shows an annotation of the interaction, highlighting how the construction and adaptation of the behavior plan is achieved using our BMLA extensions. First a behavior plan, consisting of several concatenated BML blocks is constructed (1). As an acknowledgment of the user is detected, the agent responds with a head nod (using BML block `nod1`) and continues speaking (2). Once a cough (or other noise) is detected, the system interrupts the running BML block(s) (here using BML block `remove`) and holds the execution of the next block, by prepending a pause BML block in front of it (3). The pause block is realized using a preplanned BML block (see also Table 1). While the user is still coughing, the agent plans a possible continuation to be executed after the cough (4). This continuation is an alternative to `bml4` and is inserted in between the pause block and `bml5`. As the end of the cough is detected, the agent instantly makes use of the already planned continuation by simply activating (starting) the pause behavior (5). Another user acknowledgement is met by a head nod in (6).

In (7), a typical situation in incremental processing is encountered. A user interrupt is first wrongly detected as an acknowledgement and met by a head nod. The incremental detector later adjusts its classification of this user response to a stop request, which immediately interrupts the ongoing behavior. The agent then asks the user what is going on (8). This is realized using BML blocks `interrupt1` and `breakup`, respectively.

6 Conclusion

Achieving the fluidity, responsiveness and adaptiveness of human conversational interaction is still a key challenge for ECA research, both with regard to the technical enablement of such machine abilities as well as with regard to the eventual acceptance of these systems as face-to-face conversation partners. Based on a growing body of recent work—by others as well as by ourselves—and based on an analysis of human conversational behavior and phenomena, we have argued that

[4] Here we follow the incremental feedback processing design of [27] where feedback is handled through a cascade of incremental classifiers: the first classifier of this cascade is a rapid acknowledgment/non-acknowledgment classifier, which we place in the Behavior Interpreter. Slower function classifiers (e.g., for non-understanding, non-agreement, etc.) are placed in the Function Interpreter.

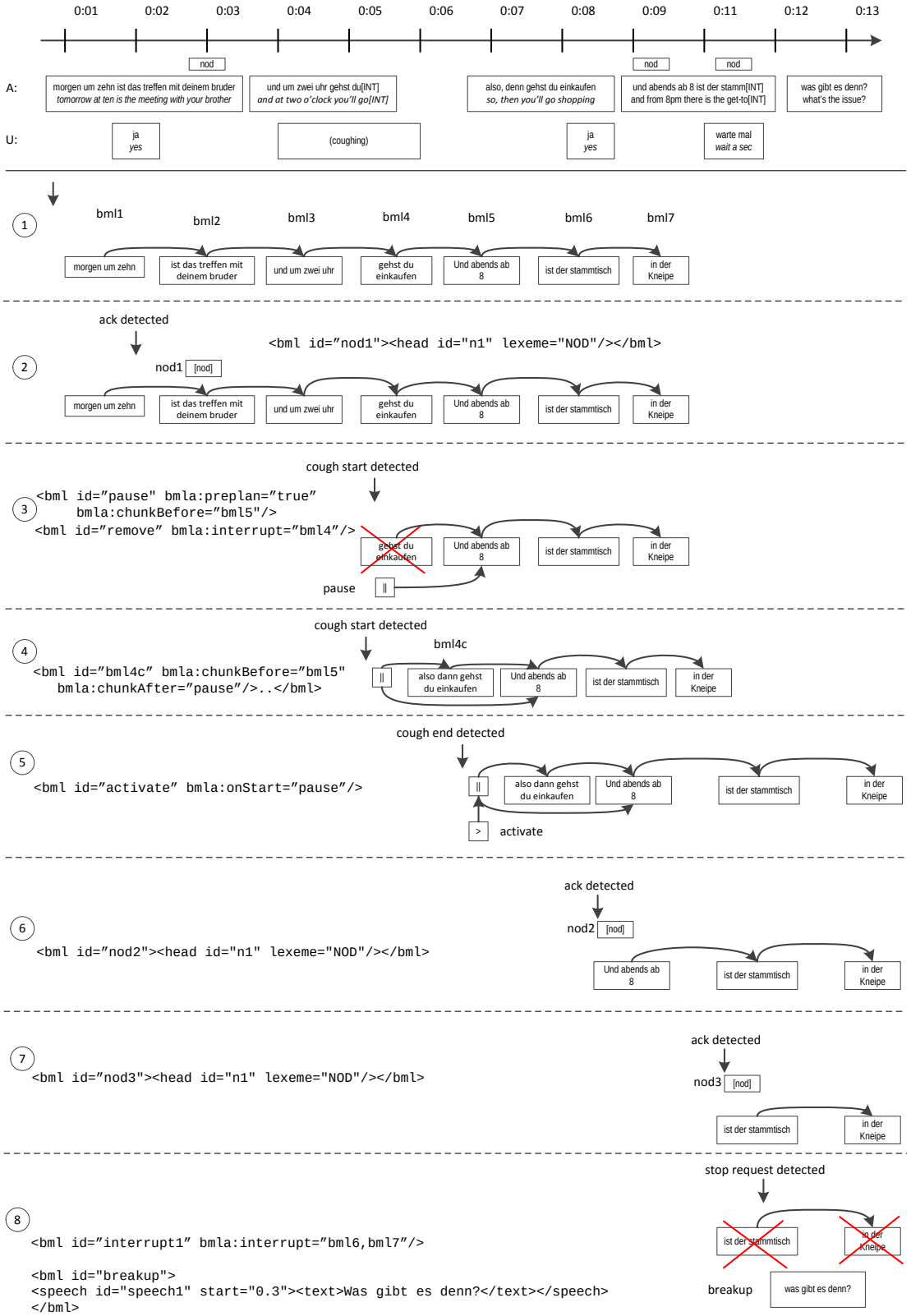

Fig. 9 Annotated example interaction.

incremental processing and a closer integration of concurrent output generation and input processing are two indispensable requirements for these qualities. Both, however, raise research questions that are relatively new to the field of ECA research and that need to be met at a basic architectural level of behavior processing.

We have presented the ASAP framework that aims to provide a unified basis for exploring those questions, both conceptually and technically. The models we have devised and the agents we have developed using this framework by bringing together several previous implementations onto a coherent architectural basis have yielded promising results that—to the best of our knowledge—exceed existing systems. Technical evaluation of the systems we have built (e.g., see [3]) reveal that incremental processing allows for faster and more fluid interactional behavior and that this behavior is perceived as significantly more natural.

It stands to reason that there are numerous open challenges and questions with this kind of architectural ECA design. For example, an incrementally steered ECA inherently may correct itself more than a non-incremental ECA. In extreme cases, this might lead to the ECA being perceived as unreliable, insecure, etc. and the interaction being less efficient. It will thus be important to develop models to assess whether input information is reliable enough to start generating output based on it (cf. the recent work on Dialogue State Tracking). Such models may be dynamic based on learning and, e.g., adjust their thresholds based on the amount of repairs in the ongoing conversation. Another issue is the question of what the best size for an incremental unit is. Theoretically, this size emerges from the internal processes that are triggered into activity by a minimal amount of characteristic input and produce output as soon as a minimal amount of output is available [13]. In practice, it is not clear what sizes at the different levels provide the most fluid and most natural conversational behavior of an ECA. Again, an approach that allows for dynamically adjusting the increment size based on both internal processing measures and external measures of dialogue success (or problems) would seem attractive. Future work will need to combine the further development of the framework with empirical evaluations, in order to clarify these questions and to endow ECAs with increasingly advanced abilities of real-time conversation.

Acknowledgements This research is supported by the Deutsche Forschungsgemeinschaft (DFG) in the Center of Excellence EXC 277 in 'Cognitive Interaction Technology' (CITEC) as well as the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster 'it's OWL', managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the content of this publication.

References

1. M. Atterer, T. Baumann, and D. Schlangen. No sooner said than done? Testing incrementality of semantic interpretations of spontaneous speech. In *Proceedings of INTERSPEECH 2009*, pages 1855–1858, Brighton, UK, 2009.
2. T. Baumann and D. Schlangen. Inpro_iSS: A component for just-in-time incremental speech synthesis. In *Proceedings of the ACL System Demonstrations*, pages 103–108, Jeju Island, Korea, 2012.
3. H. Buschmeier, T. Baumann, B. Dosch, S. Kopp, and D. Schlangen. Combining incremental language generation and incremental speech synthesis for adaptive information presentation. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 295–303, Seoul, South Korea, 2012.
4. H. Buschmeier and S. Kopp. 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, 2011.
5. O. Buss and D. Schlangen. DIUM – An incremental dialogue manager that can produce self-corrections. In *SemDial 2011: Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue*, pages 47–54, Los Angeles, CA, 2011.
6. J. Cassell, T. Bickmore, L. Campbell, H. Vilhjálmsson, and H. Yan. Human conversation as a systems framework: Designing Embodied Conversational Agents. In J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, editors, *Embodied Conversational Agents*, pages 29–63. The MIT Press, Cambridge, MA, 2000.
7. H. H. Clark and M. A. Krych. Speaking while monitoring addressees for understanding. *Journal of Memory and Language*, 50:62–81, 2004.
8. N. Crook, D. Field, C. Smith, S. Harding, S. Pulman, M. Cavazza, D. Charlton, R. Moore, and J. Boye. Generating context-sensitive ECA responses to user barge-in interruptions. *Journal on Multimodal User Interfaces*, 6:13–25, 2012.
9. I. de Kok and D. Heylen. Integrating backchannel prediction models into embodied conversational agents. In *Proceedings of the 12th International Conference on Intelligent Virtual Agents*, pages 268–274, Santa Cruz, CA, 2012.
10. F. Eyben, M. Woellmer, and B. Schuller. openSMILE – the Munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th International Conference on Multimedia*, pages 1459–1462, Florence, Italy, 2010.
11. S. Garrod and M. J. Pickering. Why is conversation so easy? *Trends in Cognitive Sciences*, 8:8–11, 2004.
12. A. Gravano and J. Hirschberg. Turn-taking cues in task-oriented dialogue. *Computer Speech and Language*, 25:601–634, 2011.
13. M. Guhe. *Incremental Conceptualization for Language Production*. Lawrence Erlbaum Associates, Mahwah, NJ, 2007.
14. P. Haazebroek, S. van Dantzig, and B. Hommel. A computational model of perception and action for cognitive robotics. *Cognitive Processing*, 12:355–365, 2011.
15. A. Hartholt, D. Traum, S. C. Marsella, A. Shapiro, G. Stratou, and A. Leuski. All together now. In *Proceedings of the 13th International Conference on Intelligent Virtual Agents*, pages 368–381, Edinburgh, UK, 2013.
16. B. Hartmann, M. Mancini, and C. Pelachaud. Formational parameters and adaptive prototype instantiation for MPEG-4 compliant gesture synthesis. In *Computer Animation*, pages 111–119, 2002.
17. G. Hoffman and C. Breazeal. Anticipatory perceptual simulation for human-robot joint practice: Theory and application study. In *Proceedings of the 23rd AAAI Confererence for Artificial Intelligence*, pages 1357–1362, Chicago, Illinois, 2008.
18. H. Hoffmann. Perception through visuomotor anticipation in a mobile robot. *Neural Networks*, 20:22–33, 2007.
19. C. Howes, M. Purver, P. G. T. Healey, G. Mills, and E. Gregoromichelaki. On incrementality in dialogue: Evidence from compound contributions. *Dialogue & Discourse*, 2:297–311, 2011.

20. P. G. Kenny, T. D. Parsons, C. Pataki, M. Pato, C. St. George, J. Sugar, and A. Rizzo. Virtual Justina: A PTSD virtual patient for clinical classroom training. *Annual Review of CyberTherapy and Telemedicine*, 6:113–118, 2008.
21. S. Kopp. Social resonance and embodied coordination in face-to-face conversation with artificial interlocutors. *Speech Communication*, 52:587–597, 2010.
22. S. Kopp, L. Gesellensetter, N. C. Kramer, and I. Wachsmuth. A conversational agent as museum guide – Design and evaluation of a real-world application. In *Proceedings of the 5th International Working Conference on Intelligent Virtual Agents*, pages 329–343, Kos, Greece, 2005.
23. S. Kopp, B. Krenn, S. C. Marsella, A. N. Marshall, C. Pelachaud, H. Pirker, K. R. Thórisson, and H. H. Vilhjálmsson. Towards a common framework for multimodal generation: The behavior markup language. In *Proceedings of the 6th International Working Conference on Intelligent Virtual Agents*, volume 4133, pages 205–217, Marina del Rey, CA, 2006.
24. S. Kopp and I. Wachsmuth. Synthesizing multimodal utterances for conversational agents. *Computer Animation and Virtual Worlds*, 15:39–52, 2004.
25. O. Lemon and A. Gruenstein. Multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer–Human Interaction*, 11:241–267, 2004.
26. P. Lison and G.-J. Kruijff. Salience-driven contextual priming of speech recognition for human-robot interaction. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 636–640, Patras, Greece, 2008.
27. D. Neiberg and K. P. Truong. Online detection of vocal listener responses with maximum latency constraints. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 5836–2539, 2011.
28. A. Nijholt, D. Reidsma, H. van Welbergen, H. op den Akker, and Z. M. Ruttkay. Mutually coordinated anticipatory multimodal interaction. In A. Esposito, N. G. Bourbakis, N. Avouris, and I. Hatzilygeroudis, editors, *Verbal and Nonverbal Features of Human-Human and Human-Machine Interaction*, pages 70–89, Berlin, Germany, 2008. Springer Verlag.
29. D. Reidsma, E. Dehling, H. van Welbergen, J. Zwiers, and A. Nijholt. Leading and following with a virtual trainer. In *Proceedings of the 4th International Workshop on Whole Body Interaction in Games and Entertainment*, Lisbon, Portugal, 2011.
30. D. Reidsma, A. Nijholt, and P. Bos. Temporal interaction between an artificial orchestra conductor and human musicians. *Computers in Entertainment*, 6:1–22, 2008.
31. D. Reidsma, H. van Welbergen, R. Poppe, P. Bos, and A. Nijholt. Towards bi-directional dancing interaction. In *Proceedings of the 5th International Conference on Entertainment Computing*, pages 1–12, Cambridge, UK, 2006.
32. T. Ribeiro, M. Vala, and A. Paiva. Thalamus: Closing the mind-body loop in interactive embodied characters. In *Proceedings of the 12th International Conference on Intelligent Virtual Agents*, pages 189–195, Santa Cruz, CA, 2012.
33. H. Sacks, E. A. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50:696–735, 1974.
34. A. Sadeghipour and S. Kopp. Embodied gesture processing: Motor-based perception–action integration in social artificial agents. *Cognitive Computation*, 3:419–435, 2011.
35. E. Schegloff. Overlapping talk and the organization of turn-taking for conversation. *Language in Society*, 29:1–63, 2000.
36. S. Scherer, S. Marsella, G. Stratou, Y. Xu, F. Morbini, A. Egan, A. S. Rizzo, and L.-P. Morency. Perception Markup Language: Towards a standardized representation of perceived nonverbal behaviors. In *In Proceedings of the 12th International Conference on Intelligent Virtual Agents*, pages 455–463, Santa Cruz, CA, 2012.
37. D. Schlangen, T. Baumann, H. Buschmeier, O. Buß, S. Kopp, G. Skantze, and R. Yaghoubzadeh. Middleware for incremental processing in conversational agents. In *Proceedings of the 11th Annual SIGdial Meeting on Discourse and Dialogue*, pages 51–54, 2010.
38. D. Schlangen and G. Skantze. A general, abstract model of incremental dialogue processing. *Dialogue & Discourse*, 2:83—111, 2011.
39. W. Schuler, S. Wu, and L. Schwartz. A framework for fast incremental interpretation during speech decoding. *Computational Linguistics*, 35:313–343, 2009.
40. S. Seneff, C. Wang, L. Hetherington, and G. Chung. A dynamic vocabulary spoken dialogue interface. In *Proceedings of INTERSPEECH 2004*, pages 321–324, Jeju Island, Korea, 2004.
41. G. Skantze and A. Hjalmarsson. Towards incremental speech generation in dialogue systems. In *Proceedings of the 11th Annual SIGdial Meeting on Discourse and Dialogue*, pages 1–8, 2010.
42. M. Stone, C. Doran, B. Webber, T. Bleam, and M. Palmer. Microplanning with communicative intentions: The SPUD system. *Computational Intelligence*, 19:311–381, 2003.
43. R. L. Street. Speech convergence and speech evaluation in fact-finding interviews. *Human Communication Research*, 11:139–169, 1984.
44. M. K. Tanenhaus, M. J. Spivey-Knowlton, K. M. Eberhard, and J. C. Sedivy. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634, 1995.
45. K. R. Thórisson. *Communicative Humanoids. A Computational Model of Psychosocial Dialogue Skills*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1996.
46. D. Traum, D. DeVault, J. Lee, Z. Wang, and S. Marsella. Incremental dialogue understanding and feedback for multiparty, multimodal conversation. In *Proceedings of the 12th International Conference on Intelligent Virtual Agents*, pages 275–288, Santa Cruz, CA, 2012.
47. H. van Welbergen, D. Reidsma, and S. Kopp. An incremental multimodal realizer for behavior co-articulation and coordination. In *Proceedings of the 12th International Conference on Intelligent Virtual Agents*, pages 175–188, Santa Cruz, CA, 2012.
48. H. H. Vilhjálmsson, N. Cantelmo, J. Cassell, N. E. Chafai, M. Kipp, S. Kopp, M. Mancini, S. C. Marsella, A. N. Marshall, C. Pelachaud, Z. M. Ruttkay, K. R. Thórisson, H. van Welbergen, and R. J. van der Werf. The Behavior Markup Language: Recent developments and challenges. In *Proceedings of the 7th International Conference on Intelligent Virtual Agents*, pages 99–120, Paris, France, 2007.
49. A. Wykowska, A. Schubö, and B. Hommel. How you move is what you see: action planning biases selection in visual search. *Journal of Experimental Psychology: Human Perception and Performance*, 35:1755–1769, 2009.