A Demonstration of Continuous Interaction with Elckerlyc

Herwin van Welbergen, Dennis Reidsma, Job Zwiers
Human Media Interaction
University of Twente, the Netherlands
{welberge|dennisr|zwiers}@ewi.utwente.nl

Abstract

We discuss behavior planning in the style of the SAIBA framework for continuous (as opposed to turn-based) interaction. Such interaction requires the real-time application of minor shape or timing modifications of running behavior and anticipation of behavior of a (human) interaction partner. We discuss how behavior (re)planning and on-the-fly parameter modification fit into the current SAIBA framework, and what type of language or architecture extensions might be necessary. Our BML realizer Elckerlyc provides flexible mechanisms for both the specification and the execution of modifications to running behavior. We show how these mechanisms are used in a virtual trainer and two turn taking scenarios.

Keywords: Continuous interaction, SAIBA, BML realization, virtual humans

1 Introduction

Virtual humans often interact with users using a combination of speech with gestures in a conversational setting. They tended to be developed using a turn-based interaction paradigm, but this is changing towards a *continuous* interaction paradigm, where actors perceive acts and speech of others continuously, and where actors can act continuously, simultaneously and therefore overlapping in time (Nijholt et al., 2008). This raises the question how acting has to be planned; clearly the traditional "perceive-plan-act" cycle from agent theories does not apply here. Still, it is clear that (human as well as virtual) actors *do* perform some form of "action planning", both for the long term as well as for the short term. Behavior of interaction partners (real or virtual) is dealt with by, on the one hand, *predicting* such behavior and, on the other hand, *re-planning* and *modifying* the virtual human's own behavior. There are various forms of this revision process, some more disruptive than others. As an example, consider a fitness trainer that has just identified a problem in that the group he is coaching starts "lagging" and does no longer follow the correct tempo of their exercise. A disruptive revision would be to stop the exercise, explain what went wrong, and to start over again. Although this is certainly a possible solution, we propose a more subtle, preferred, approach, where the trainer starts moving a little faster, and a little ahead of the group in order to try to speed up the tempo of the group. Within the SAIBA framework for behavior planning, shown in Figure 2, the first approach requires a revision of intents, followed by re-planning speech and bodily behavior.

Our alternative approach circumvents this and applies a more direct revision of bodily behavior, based upon (short term) prediction by means of so called *Anticipators*, combined with corrective adjustments of already ongoing behavior. This leads to a flexible planning approach in which part of the planning can be done beforehand, and part has to be done "on the fly". In the latter case, parts of the behavior have been executed already, and other parts can still be modified. We focus on the specification (both of the plan itself and of changes to the plan) and execution of such flexible plans. We provide abstractions for the prediction of sensor input and show how we can synchronize our output to these predictions in a flexible manner. To demonstrate the feasibility of the multimodal output generation part of our system without having to invest a lot of work in the sensing part, we have implemented placeholders for the predictors.

In this paper, we present several scenarios in which continuous interaction is achieved using small adjustments in the timing and shape of behaviors (for example: gesture or speech) while it is being executed by a virtual human. We show how such small adjustments can be specified and how we implemented these behaviors in our behavior realizer Elckerlyc. We intend to demonstrate our implementation in the demo session of the workshop.

2 Elckerlyc's Architecture

We base our architecture (see Figure 2) on the SAIBA Framework (Kopp et al., 2006), which contains a three-stage process: *communicative intent planning*, multimodal *behavior planning*, resulting in a BML stream, and *behavior realization* of this stream. Elckerlyc encompasses the realization stage. It takes a specification of the intended behavior of a virtual human written in the Behavior Markup Language (BML) (Kopp et al., 2006) and executes this behavior through the virtual human. The BML stream contains behaviors (such as speech, gesture, head movement etc.) and specifies how these behaviors are synchronized. Synchronization of the behaviors to each other is done through BML *constraints* that link synchronization points in one behavior (start, end, stroke, etc; see also Figure 1) to synchronization points in another behavior. BML can be used to add new behaviors or remove running behaviors, but does not contain mechanisms to slightly modify behavior that is already running. However, we argue that some desired changes to planned behavior are only on their timing or parameter values (speak louder, increase gesture amplitude) and should not lead to completely rebuilding the animation or speech plan. Such small adaptations of the timing of or shape of planned behavior occur in conversations and other interactions (Nijholt et al., 2008).

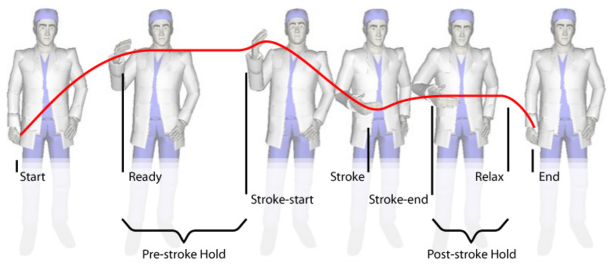


Figure 1: Standard BML synchronization points (picture from `http://wiki.mindmakers.org/projects:bml:main`)

There is typically a planning delay between sending a BML stream to the Behavior Realizer (Elckerlyc in our case) and the realization of this stream. By fine-tuning running or planned behaviors rather than re-planning the complete behavior plan when only small changes are required, we avoid this planning delay and allow fluent and timely behavior execution. Others have used similar mechanisms for incremental planning in gesture/speech synthesis: Kopp and Wachsmuth (2004) make use of an incremental planning mechanisms that allows the late planning of transitions between segments of gesture and speech, which are highly context dependent (depending on current gesture and the next gesture), but for which some parts can be pre-planned (e.g. the speech synthesis). In human-human behavior, there is some evidence of similar pre-planning mechanisms (Nijholt et al., 2008), for example to allow rapid overlaps between turns in a dialog. We extend BML to allow the specification of synchronization to anticipated timing of external events (from the environment, or other (virtual) humans). Elckerlyc allows partial pre-planning of behavior that is timed to such events. The timing of such behavior is refined and completed continuously, while keeping inter-behavior constraints consistent.

To achieve this incremental temporal control, we introduced *Time Pegs* and *Anticipators*. BML specifies constraints between behaviors, indicating that their synchronization points should occur at the same time. We maintain a list of Time Pegs – symbolically linked to those synchronization points that are constrained to be on the same time – on the *Peg Board*, together with the current

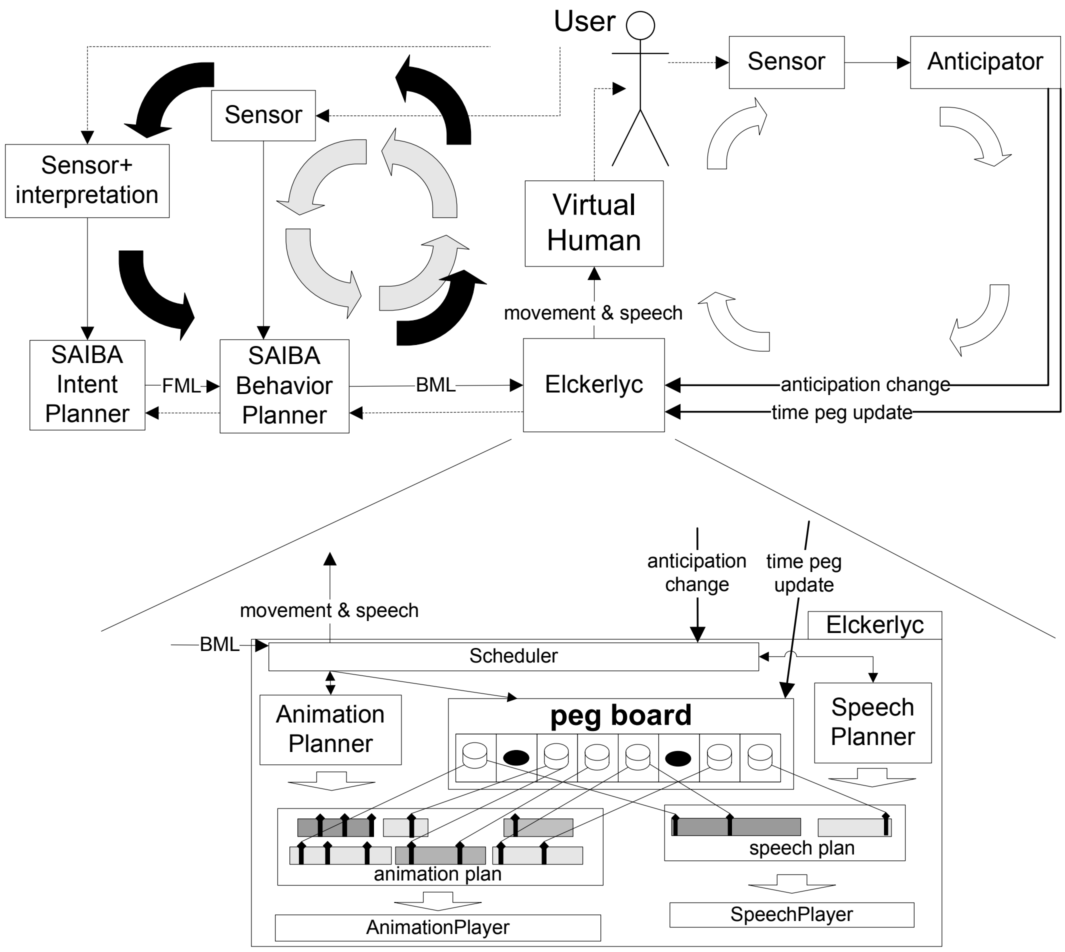
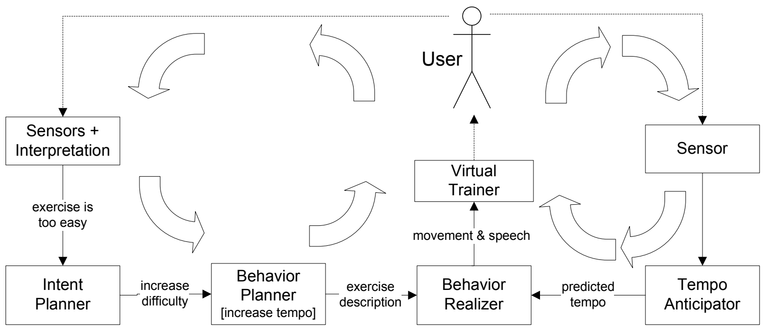


Figure 2: Elckerlyc architecture and its location within the SAIBA framework

expectation of their actual execution time (which may change at a later time and can be unknown).

Interaction with the world – and conversation partner – is achieved through Anticipators. An Anticipator instantiates Time Pegs that can be used in the BML stream to constrain the timing of behaviors. This is specified in a similar manner as BML constraints: a synchronization point of a behavior is linked to the synchronization point of an Anticipator (as identified by the Anticipator id and its TimePeg id, see Figure 3, 4 for some examples). The Anticipator uses sensors that perceive events in the real world to continuously update the Time Pegs, by extrapolating the perceptions into predictions of the timing of future events.

Several feedback loops between user and agent behavior can exist in the SAIBA framework. The SAIBA Intent Planner makes use of *interpreted* user behavior to decide on the Intent of actions that are to be executed by the virtual human (indicated with the black arrows in Figure 2). Bevacqua et al. (2009) argue for another feedback loop (indicated with the gray arrows), using sensor-activated unconscious and unintentional (so not originating from the Intent Planner) behavior in the Behavior Planner. One example of such behavior is mimicry, which they propose to implement by submitting new BML to the Realizer, which then has to be re-planned. In this paper we demonstrate the need for an even tighter feedback loop (indicated with the white arrows) which allows small modifications based on user observations to be made to running behaviors directly, without the need for re-planning behavior. Similar layered feedback loops between a user and a virtual human occur in the Ymir system (Thórisson, 2002).



```
<bml id="bml1">
  <bmlt:procanimation id="exercise1" name="squat-jump"/>
  <constraint id="c1">
    <synchronize ref="exercise1:squatdown">
      <sync ref="exerciseAnticipator:squatdown-0.5"/>
    </synchronize>
    <synchronize ref="exercise1:jumpstart">
      <sync ref="exerciseAnticipator:jumpstart-0.5"/>
    </synchronize>
    ...
  </constraint>
</bml>
```

Figure 3: Exercise scenario. `exercise1:squatdown` and `exercise1:jumpstart` refer to the squated down position and the start of the jump in the squat-jump exercise animation respectively. `exerciseAnticipator:squatdown` and `exerciseAnticipator:jumpstart` refer to the anticipated timing of squatdown and jumpstart as predicted by movement of the user.

Elckerlyc can be used as a black box that converts BML into multi-modal behavior for a VH.[1] If required however, direct access to the Scheduler, Planners, Plans and Players is also available. Some of this functionality is used in the demo scenarios described in this paper to adapt the parameter values of ongoing behavior (e.g. speak louder). We refer the reader to (van Welbergen et al., 2010) for a extensive explanation of Elckerlyc's architecture.

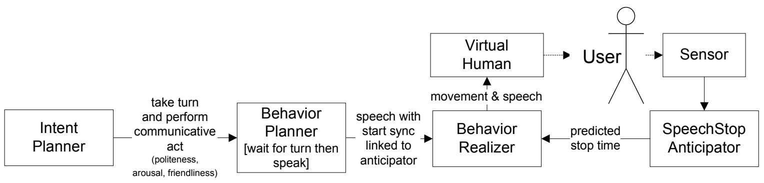
3 Scenarios

3.1 Guiding Exercise Tempo

A virtual (fitness) trainer executes an exercise together with a human user in a certain tempo. The trainer would like to increase the tempo that the user is moving in. A subtle technique to achieve this is to move in the same tempo as the user but slightly ahead of him, so he constantly has the feeling of being 'too late' in his movements (a similar technique is used by our virtual conductor to guide the tempo of a real orchestra (Reidsma et al., 2008)). We assume that an Anticipator can be designed that can perceive the tempo a user is exercising in and from this information extrapolates future exercise time events [2]. By making use of the time predictions of this Anticipator, we can specify the trainer's movement to be slightly ahead of them. Note how the availability of a specific Anticipator, and its exact implementation, are application dependent.

[1]This functionality will be shown in our demo and can be tested from the Elckerlyc webstart at `http://hmi.ewi.utwente.nl/showcase/Elckerlyc`.

[2]In our demo we fake these perceptions by using space bar presses instead. For simple fitness exercises one could use, e.g., accelerometers attached to the wrists and ankles of the user, detecting the tempo from the peak structure in the accelerations. Future peak points are then predicted by extrapolating the average tempo of the last few peak points in the exercise performed by the user.


Intent Planner
take turn and perform communicative act (politeness, arousal, friendliness)
Behavior Planner [wait for turn then speak]
speech with start sync linked to anticipator
Behavior Realizer
movement & speech
Virtual Human
User
Sensor
SpeechStop Anticipator
predicted stop time

```
<bml id="bml1">
  <speech id="speech1" start="speechStopAnticipator:stop+x">
    <text>Bla bla</text>
  </speech>
</bml>
```

Figure 4: Taking the turn

Figure 3 illustrates this scenario. From interpretation of sensor values (for example: heart rate), the Intent Planner is informed that the current exercise is too easy. It decides to increase the exercise difficulty. The Behavior Planner selects a strategy to achieve this: it decides to gradually increase the tempo of the exercise. This is realized using the strategy described above. This strategy is encoded in the BML block shown in Figure 3. This block describes how synchronization points of a procedural exercise animation (`exercise1`) are synchronized to be slightly ahead (0.5 seconds) of the anticipated synchronization points in the exercise as executed by the user. Each of these synchronization points is linked to a Time Peg. The timing of these Time Pegs is continuously updated using the perceived tempo of the user in the feedback loop on the right, so that the trainer keeps on moving ahead of the user, even when the tempo of the user changes. Of course, if the tempo of the user deviates really too much from the desired tempo, the Intent Planner might still decide on a different exercise strategy, such as choosing a completely different exercise.

3.2 Turn Taking in Speech

3.2.1 Taking the Turn

Humans can take the turn at different moments, for example, slightly before their interaction partner stops speaking, at exactly the moment their interaction partner stops speaking, or slightly after their interaction partner stops speaking. The turn taking strategy used can modulate the impression of politeness, friendliness and arousal of the virtual human (ter Maat and Heylen, 2009). We assume that we can design an anticipator that can predict the end of speech of a user[3], called the `speechStopAnticipator`. Figure 4 illustrates a turn taking scenario. The Intent Planner decides to take the turn and perform a communicative act. The Behavior Planner selects a turn taking strategy, based on the politeness, arousal and friendliness of the virtual human. In the illustrated case, it waits for the user to stop speaking and starts speaking after a certain delay x (could be negative to start speaking slightly before the user stops speaking). To allow an immediate response of the virtual human to the (anticipated) speech stop of a user, the behavior is pre-planned, and its start time is synchronized to this (can be currently unknown) anticipated speech stop. The Behavior Planner thus only specifies that the virtual human starts speaking after the user stops speaking, and the exact and precisely timed execution of this behavior is handled by the Behavior Realizer, using the speech stop anticipator.

3.2.2 Keeping the Turn

To keep the turn, one can simply ignore the interruption request of the interaction partner. Alternatively, one can raise the volume of the voice at the moment of the interrupting speech. The turn keeping strategy used can modulate the impression of friendliness and arousal of the virtual human (ter Maat and Heylen, 2009). Raising the voice requires a real-time change in parameter

[3]We currently fake the detection of speech endings by pressing the space bar.

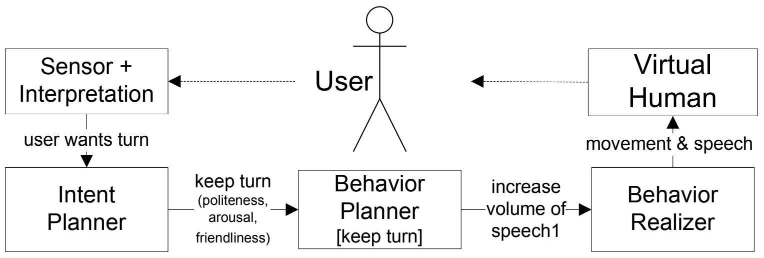


Figure 5: Keeping the turn

```
<bml id="bml2" scheduling="tight-merge"/>
  <bmlt:setparameter id="reparam1" start="10" end="speech1:end"
    target="speech1" parameter="volume" curve="linear" startvalue="25" endvalue="100"/>
</bml>
```

Figure 6: Change the volume of `speech1`, starting at absolute time 10, until `speech1:end`. The volume ranges from 25 to 100.

values (volume in this case) of the virtual human's speech. Elckerlyc currently allows this by providing direct access to the parameter values of each behavior in an animation or speech plan.

Figure 5 illustrates this scenario. The Intent Planner is informed by an interpretation of sensor values that the user would like to get the turn [3]. The Intent Planner decides that the virtual human would like to keep the turn. Based on the provided politeness, arousal and friendliness values, the Behavior Planner decides to realize this in intent by increasing the volume of behavior `speech1`. Currently this is achieved by an ad hoc function call in the Behavior Realizer.

4 Discussion

We have shown that Elckerlyc's Anticipators and Time Pegs provide a flexible formalism for both the specification and the execution of behavior that requires anticipation of the behavior of a (human) interaction partner. They also provide a flexible pre-planning mechanism for behaviors that have to be executed at a (to be determined) later time moment.

We have discussed a scenario in which a parameter value change of running behavior is desired in Section 3.2.2. Currently we apply such parameter value changes in a ad hoc manner. We change the parameter values of a motion or speech fragment by accessing the animation plan or speech plan directly and adapting the parameters of the (possibly running) behavior. We then need to take care of the parameter value curve and the duration of the parameter value change as well. We are currently exploring more formal methods of parameter change specification and execution. An interesting method to achieve this is implemented in the Multimodal Presentation Markup Language (Brügmann et al., 2008): parameter value changes are implemented as an Action (a concept similar to a BML behavior). This allows one to easily define parameter value changes outside the Realizer and to specify the synchronization of the change to other behaviors in a conceptually similar manner as behavior synchronization. Additionally, such a script based specification of parameter value changes allows easy experimentation with parameter values and curves. Figure 6 shows how such a parameter value change could be expressed using BML [4].

However, parameter value change as a BML behavior does not match very well with the other behavior types (gaze, locomotion, speech, etc.) and requires specialized planning mechanisms to be able to refer to BML elements from previously planned BML blocks. Furthermore, we probably do not want parameter values to modify synchronization constraints like other behaviors can do, they simply need to adhere to the timing prescribed by other behaviors. So conceptually it might

[4]Note that this BML block requires a special scheduling mechanism (tight-merge) to allow it to refer to a behavior in a previous BML block, see `http://wiki.mindmakers.org/projects:bml:multipleblockissue`

be nicer to provide a separate (non BML) channel in the Realizer through which a specification of parameter value changes (the timing of which can depend on timing of BML behaviors and that can target a specific BML behavior) can be sent.

Acknowledgements

Many of the ideas and issues discussed here emerged during our very interactive talk and visit of Stefan Kopp's Sociable Agents Group in Bielefeld, we would like thank the participants for a fruitful discussion on continuous interaction. We thank Bart van Straalen for his help in refining the Intent/Behavior planning terminology and Mark ter Maat for explaining his insights on turn taking strategies. This research has been supported by the GATE project, funded by the Dutch Organization for Scientific Research (NWO) and the Dutch ICT Research and Innovation Authority (ICT Regie).

References

Bevacqua, E., Prepin, E., de Sevin, R., Niewiadomski, R., and Pelachaud, C. (2009). Reactive behaviors in SAIBA architecture. In *Towards a Standars Markup Language for Embodied Dialogue Acts Workshop at Autonomous Agents and Multi-Agent Systems.*

Brügmann, K., Dohrn, H., Prendinger, H., Stamminger, M., and Ishizuka, M. (2008). Phase-based gesture motion parametrization and transitions for conversational agents with MPML3D. In *INtelligent TEchnologies for interactive enterTAINment*, pages 1–6. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Kopp, S., Krenn, B., Marsella, S., Marshall, A. N., Pelachaud, C., Pirker, H., Thórisson, K. R., and Vilhjálmsson, H. H. (2006). Towards a common framework for multimodal generation: The behavior markup language. In *Intelligent Virtual Agents*, volume 4133 of *LNCS*, pages 205–217. Springer.

Kopp, S. and Wachsmuth, I. (2004). Synthesizing multimodal utterances for conversational agents. *Comput. Animat. Virtual Worlds*, 15(1):39–52.

Nijholt, A., Reidsma, D., van Welbergen, H., op den Akker, H. J. A., and Ruttkay, Z. M. (2008). Mutually coordinated anticipatory multimodal interaction. In *Nonverbal Features of Human-Human and Human-Machine Interaction*, volume 5042 of *LNCS*, pages 70–89. Springer.

Reidsma, D., Nijholt, A., and Bos, P. (2008). Temporal interaction between an artificial orchestra conductor and human musicians. *Computers in Entertainment*, 6(4):1–22.

ter Maat, M. and Heylen, D. (2009). Turn management or impression management? In *Intelligent Virtual Agents*, volume 5773 of *LNCS*, pages 467–473. Springer Verlag.

Thórisson, K. R. (2002). Natural turn-taking needs no manual: Computational theory and model, from perception to action. In *Multimodality in Language and Speech Systems*, pages 173–207. Kluwer Academic Publishers, Dordrecht, The Netherlands.

van Welbergen, H., Reidsma, D., Ruttkay, Z. M., and Zwiers, J. (2010). Elckerlyc: A BML realizer for continuous, multimodal interaction with a virtual human. *To Appear in Journal on Multimodal User Interfaces.*