Models for Decision Making in Video Mediated Communication

Wolfgang Weiss
Institute for Information and Communication Technologies
JOANNEUM RESEARCH
Graz, Austria
wolfgang.weiss@joanneum.at

Manolis Falelakis
Department of Computing
Goldsmiths College
University of London
London, UK
m.falelakis@gold.ac.uk

Rene Kaiser
Institute for Information and Communication Technologies
JOANNEUM RESEARCH
Graz, Austria
rene.kaiser@joanneum.at

Marian F. Ursu
Department of Theatre, Film and Television
University of York
York, UK
marian.ursu@york.ac.uk

ABSTRACT

There is an abundance of players in the domain of video communication, offering systems for various purposes and different capabilities. However, these systems are still very limited in adapting the visual presentation of the remote participants to the participants' needs.

In this paper we describe the components, the implementation and the overall behaviour of an automatic decision-making system for video communication, known as Orchestration Engine. This system analyses the communication behaviour of the participants and applies TV-like mixing grammars adapt the user interface accordingly. The behaviour is implemented as a set of rules, divided into different levels and fulfilling aesthetic and pragmatic requirements. This rule set is defined declaratively, facilitating the adjustments/modifications in desired behaviour, while proving fast enough for real-time decision making.

Categories and Subject Descriptors

H.5.m [**Information Interfaces and Presentation (e.g. HCI)**]: Miscellaneous

Keywords

Videoconferencing; communication orchestration; automatic decision making; video mediated group communication

1. INTRODUCTION

Audio-visual communication pervades slowly but continuously our daily life, mainly driven by the availability of broadband services and mobile devices. One important aspect in our life is social communication with our friends. This type of communication is characterised by certain features, for example people can join and leave at any point, the dynamic of the conversation might change over time, and the network capabilities might change.

In contrast to business videoconferencing systems with dedicated hardware and special equipment, the requirements for social communication are different. Users want to use their own devices such as smartphones, tablets or personal computers which all have different properties in e.g. screen size, computing power and network capabilities. The available systems improved over the last years, but their functionality is still at the beginning, e.g. they have limited intelligence to adapt to specific communication situations. Another inherent problem implied by such video communication setups is to select from the multitude for available sources for each participant, i.e. when there are n participants and each is equipped with 1 camera, $n-1$ exterior video streams are possible candidates. An intuitive, but not always scalable option, is to show each user all video streams side by side on one screen.

The question for this work is: how can an intelligent video communication system be implemented to overcome the aforementioned problems? We will argue to analyse the communication behaviour of the participants, to apply TV like mixing grammar, and to adapt the user interface accordingly will be a step in the right direction. We will describe the components, the implementation and the overall behaviour of a decision making system, known as Orchestration Engine, developed for social communication.

The Vconect[1] project investigates novel ways of supporting mediated audio-visual communication for ad-hoc groups. We set out to investigate more sophisticated solutions with the aim of achieving better communication support through intelligent camera selection. A component which automatically executes a mixing process of different video streams is known as a Virtual Director [5]. In the realm of communication this is mostly referred to as Orchestration [8].

[1]http://www.vconect-project.eu/

UM3I'14, November 16, 2014, Istanbul, Turkey.

ACM 978-1-4503-0652-2/14/11 ...$15.00.
http://dx.doi.org/10.1145/2666242.2666250.

The following Section describes what Orchestration is, what it should consider and how it could work. Section 3 discusses the behaviour and the implementation of each component of the developed Orchestration Engine. The pros, cons and limitations of our approach are discussed in Section 4 and, finally, Section 5 concludes this work.

2. ORCHESTRATION

Communication Orchestration or simply *Orchestration* can be regarded as the decision making which controls the mixing process of all available audio and video streams. It is similar to directing a live TV transmission but in the case of video conferencing it has to address communication rather than narrative needs. Orchestration is a reasoning process which operates in real-time, individually for each participant, screen, or location participating in the communication, deciding at each point what each participant is going to see and hear from the multitude of available audio and video sources. It mediates the communication and therefore has to create a meaningful and neutral communication experience. From a technical viewpoint, the orchestration process builds upon the audio and video processing infrastructure and executes camera control and audio-visual composition (cf. [8]).

The automatic decision making process of an Orchestrator is inherently (directly or indirectly) driven by the events occurring within the participating locations. What people have been doing, who has been talking to whom, what is being said, at which pace together with non-verbal cues such as gesticulation, nodding, eye-gaze are all of potential interest here. A collection of these events (if seen from a more macroscopic temporal viewpoint) constitutes the state or context of the interaction at each time. An Orchestrator needs to receive, fuse and interpret low-level cues, coming from audio-visual and/or other sensors, in order to infer and understand the context of the interaction. Based on such understanding, it will then be able to apply a set of primitives to drive its intelligent selection of viewpoints at each time, reaching decisions such as "show person X in the upper left corner of screen Y for Z seconds". In other words, an Orchestrator software essentially performs two discrete activities, as illustrated in Figure 1:

- Understanding: analysing low-level data from different sources and/or modalities in order to understand the context and the important events occurring during the interaction of participants.
- Reacting: using appropriate, pragmatic and aesthetic, principles in order to react by choosing the most appropriate shot/viewpoint to display at each point, in other words to represent the context of interaction on a screen.

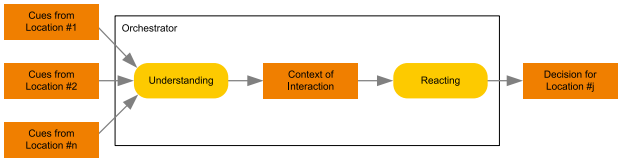

Figure 1: Partitioning the tasks of an Orchestrator.

A typical videoconferencing setup in Vconect is illustrated in Figure 2. It consists of multiple participants with multiple cameras and screens which are in locally separated locations. The Orchestration Engine is the central decision making component which executes the mixing process of the audio and video streams.

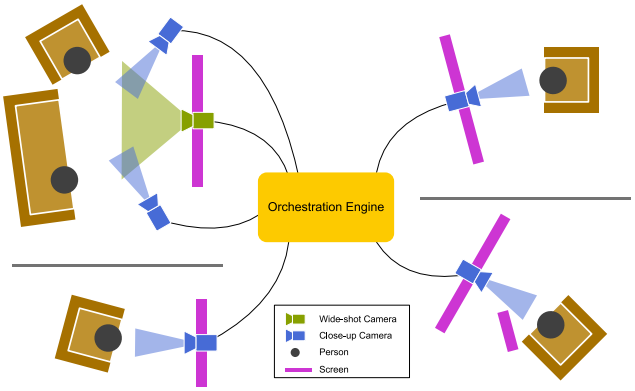

Figure 2: Schematic setup of a group videoconferencing session in bird-eye view.

3. MODELS AND IMPLEMENTATION

In this section we discuss the models and the implementation for each module of the Orchestration Engine which was developed in the Vconect project. Figure 3 illustrates the architecture with its main components. The two latter components are part of the Orchestration Engine which is a central, server-side software component.

Cue extraction: A real-time audio analysis module processes the audio stream and extracts voice activity events with a generic sound activity detector[2] indicating if the person is speaking or not. The analysis module could be easily extended to other modalities such as detecting facial expression, pitch of voice or gestures through analysing the audio and video stream.

Fusion and interpretation: Low-level cues from all locations are aggregated in the Semantic Lifting module of the Orchestration Engine. In this stage, higher-level semantic events concerning the communication as a whole, such as a "turn-shift", are generated while properties of the state of the interaction at each point, such as the "active speaker", are evaluated continuously. The module aims to achieve a computational interpretation of the current communication situation on a semantic level that can directly be evaluated by the decision making components. The Semantic Lifting module also calculates conversation metrics such as "turn shifts per active participants" based on a sliding temporal window. These metrics allow to identify e.g. the "heatedness" of a discussion.

Decision making: The application of mixing rules that result in the shot selection and the selection of the visual layout is made by the Director modules. For each

[2]https://www.itu.int/rec/T-REC-G.720.1/en

screen one separate instance reasons based on high-level events and conversation metrics received from the Semantic Lifting module. This process allows to select the optimal visual layout in combination with the necessary video streams for each user and gives best support in communication by respecting the given limitations.

3.1 Semantic Lifter

The aim of the Semantic Lifter module is to achieve a computational interpretation of the current communication situation which gives an understanding of what is going on in the conversation and therefore what is going on in front of the cameras. In other words, it models and represents the social aspects of the conversation and therefore the interaction between the participants. The output of this module allows subsequent modules to build upon this knowledge and to make meaningful decisions. The Semantic Lifter module consists of two sub components: the low-level cue lifter and the conversation metrics calculator. The former component is event-driven, meaning that it fires when someone gets the turn of a conversation or a participant talks while someone else already has the turn. On the other hand, the conversation metrics calculator, operates continuously based on sets using a sliding temporal window. It calculates metrics which represent the characteristics and attributes of the whole conversation, e.g. it ranks the participants by their level of participation.

3.1.1 Low-Level Cue Lifter

This component fuses and processes all cues which are generated by the analysis modules in each location. A cue is technically an event with a number of properties describing an actual occurrence that something has happened in a location which might be useful for further processing. The currently implemented client delivers cues about the voice activity and if a person joins or leaves a room. The low-level cue lifting component works solely on the audio information indicating if a person started or stopped to talk. It uses pattern matching to search for specific conversational patterns, generate semantically more abstract and higher-level events, and forward the latter to the subsequent components. The following higher-level events are generated by this component:

- **Turn** indicates the activity of a participant of the conversation (c.f. [3]). The move of a turn from one person to another is called a *turn-shift*. A turn start and stop can be defined as follows (resembling first order logic):
  1. $voiceActivity(START, P) \wedge \neg turn(P)$
  $\wedge \neg voiceActivity(END, P)\ after\ 600ms$
  $\rightarrow turnStart(P)$
  2. $voiceActivity(END, P) \wedge turn(P)$
  $\wedge \neg voiceActivity(START, P)\ after\ 600ms$
  $\rightarrow turnEnd(P)$

- **Cross Talk** situations are, where a person talks while another person holds the turn. This situation can be identified with following rule:
  $voiceActivity(START, P1) \wedge turn(P2)$
  $\wedge \neg voiceActivity(END, P1)\ after\ 600ms$
  $\rightarrow crossTalk(P1)$

- **Simultaneous Starts** occur easily in conferencing systems that inherently have delays generated by e.g. network latencies. Simultaneous starts are detected with following rule:
  $turnStart(P1) \wedge turnStart(P2)\ after\ 2000ms$
  $\rightarrow simultaneousStart(P1, P2)$

- **Silence** is a longer pause in a conversation which could be awkward for the participants. The rule for identifying this concept is:
  $\neg turn() \wedge \neg turnStart()\ after\ 60sec$
  $\rightarrow silence()$

3.1.2 Conversation Metrics Calculator

Conversational metrics represent the characteristics and attributes of the conversation. They provide information about how active a conversation is and who is actively participating. This component uses the output from the low-level cue lifter and calculates the metrics continuously. Let $P = \{p_k\}$ the set of participants, $T = \{\Delta t_i\}$ a set of (finely discretised) time intervals that constitute the time window w. Also, for a participant p_k during the interval Δt_i let $A_i^k = \{0, 1\}$ denote wether p_k is active during Δt_i ($A_i^k = 1$), or not ($A_i^k = 0$). In a similar fashion, $C_i^k = \{0, 1\}$ will denote whether p_k is cross talking. Currently, the following metrics are calculated and for all metrics we use a sliding time window of 60 seconds.

- **Active participation ratio** is the proportion of time that involves at least one active participant:

$$a = \frac{1}{w} \sum_{\Delta t_i \in T} \bigcup_{p_k \in P} A_i^k$$

- **Silence ratio** is defined as the proportion of time that involves no active participant i.e., essentially the dual of active participation ratio:

$$s = 1 - a$$

- **Cross talk ratio** is defined as the proportion of time that involves at least one participant cross talking:

$$c = \frac{1}{w} \sum_{\Delta t_i \in T} \bigcup_{p_k \in P} C_i^k$$

- **Number of active participants** is the total number of participants who were active within the time window.

- **Number of turn shifts** is the accumulated number of turn shift events of all participants.

- **Number of cross talks** is the accumulated number of cross talk events of all participants.

- **Number of turn shifts per participants** is the accumulated number of turn shifts of all participants divided by the number of participant.

- **Number of turn shifts per active participants** is the accumulated number of turn shifts of all participants divided by the number of participants that have been active during the time window.

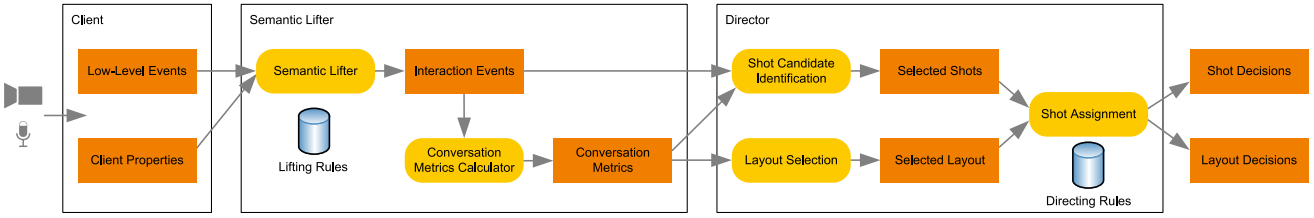

Figure 3: Architecture of the Orchestration Engine.

- **Heated discussion** states if the conversational temperature is high or not. A discussion with a high conversational temperature indicates that there is a high number of turn shifts within a short time period and that the participants might be excited. This metric evaluates to true if the *number of turn shifts per active participant* divided by the *number of active participants* is above a certain threshold.
- **Monologue** is when one person holds the turn for more than 60% of the time of the sliding time window.
- **Active participation ratio per each participant** is the active participation time (turn time) divided by the length of the sliding time window. It is calculated for each participant individually.

3.2 Director

The Director module is the decision making part of the Orchestration Engine where one instance works independently for each screen. It processes the output from the Semantic Lifter module based on a set of rules which results into shot decisions and layout decisions. This process allows to select the optimal visual layout in combination with the necessary video streams for each user independently, and to give best support in communication by respecting the given limitations.

The behaviour of this module resembles the application of mixing rules similar to those used in television. This production grammar is strongly inspired by cinematic techniques which describe methods and common conventions used in video, film and TV productions. These techniques are the building blocks and creative methods which are applied by film makers to communicate meaning, to entertain, and to evoke a particular emotional or psychological response by the audience. This includes e.g. lighting, using depth of field, focus, camera position, camera movement, framing, special effects, shot cutting effects, etc. Concepts which are applicable in the domain of video mediated communication were taken and some new concepts were introduced such as the selection of the optimal visual layouts.

The implementation of this module consists of three independent components which are described subsequently.

3.2.1 Shot Identification and Prioritisation

The main target of this component is to prioritise persons based on previously calculated conversation metrics and incoming interaction events, identify shots for each person, and to send notifications about changes in the prioritisation list. The prioritised list is created through sorting the participants list in descending order based on the *active participation ratio per each participant*. One important aspect of videoconferencing systems is to focus on the currently active person. To do so, this module identifies the most adequate person to be displayed in the main region of the selected layout based on turn shift events coming from the Semantic Lifter module. Whenever this component notices a change in the order of the prioritised list of the participants or if there is a change in the person holding the turn, it sends a notification event to the shot assignment component. Table 1 lists the editing rules implemented for this component representing pragmatic principles e.g. cutting to a person whenever it gets the turn and aesthetic principles of the Orchestration Engine e.g. making variations of shots such as to cut away to a wide shot when a temporal threshold is reached.

3.2.2 Layout Selection

The dynamics of a conversation varies over time which makes it necessary to adapt the composition of the video streams on the participants' screens. To do so, this module selects a visual layout based on events and metrics coming from the Semantic Lifter module. A layout is a predeveloped composition of 1 ... n regions laid out on the rendering surface of the screen. The regions in the layout define the size and the spatial arrangement of the video streams. Following layouts are available which are used in following situations:

- A single **full screen layout**, see also Figure 4a, which contains only one big region will be chosen when there is a monologue detected by a participant. Another reason for a single full screen layout would be if the screen size of the client is very small.
- Figure 4b illustrates a **full screen layout containing another region which** is also known as a picture-in-picture layout. This layout is be suitable when a private discussion between two people takes place. The large region is for the remote participant while the small region is used for the self-view.
- If the conversation gets more heated (animated), meaning there is a high number of turn shifts between the participants within the analysis time window, it would be beneficial for the users to see all involved participants on the screen and therefore use the **tiled layout** of Figure 4c.
- A layout, as illustrated in Figure 4d, with **focus on one person and with smaller regions** for all other participants is suitable for most situations when the screen size is big enough and the number of participants does not exceed a certain number, e.g., 10 peo-

Table 1: Editing rules in a notation resembling FOL

1.	$turnShift(P) \wedge isInOtherLocation(L, P) \rightarrow cut2CUFront(P)$
2.	$crossTalk(P) \wedge isInOtherLocation(L, P) \rightarrow cutAway2CUFront(P)$
3.	$timeSinceLast(turnShift) > threshold \wedge activePerson(P) \wedge isInOtherLocation(P) \rightarrow cut2Wide(P)$
4.	$numberOfShotsSinceLastCUOblique > threshold \wedge isInOtherLocation(PersonInFocus) \rightarrow cutAway2CUOblique$
5.	$timeSinceLastWide > 30sec \wedge isInOtherLocation(PersonInFocus) \rightarrow cutAway2Wide(PersonInfocus)$
6.	$currentShotType == CUFrontdurationOfCurrentShot > threshold \rightarrow cutAway2CueCamera$

ple, and the conversation has a low or normal pace. This is also the standard layout of the system.

- A layout, as illustrated in Figure 4e, with **two larger regions and several smaller regions in the bottom** is suitable in situations when there are two people intensively discussing while the other participants follow the conversation passively.
- If there is an intensive discussion between two people, as described for the previous layout, but the client does not allow to render more than two video stream in parallel, then a layout with only **two main regions** is suitable, see also Figure 4f.

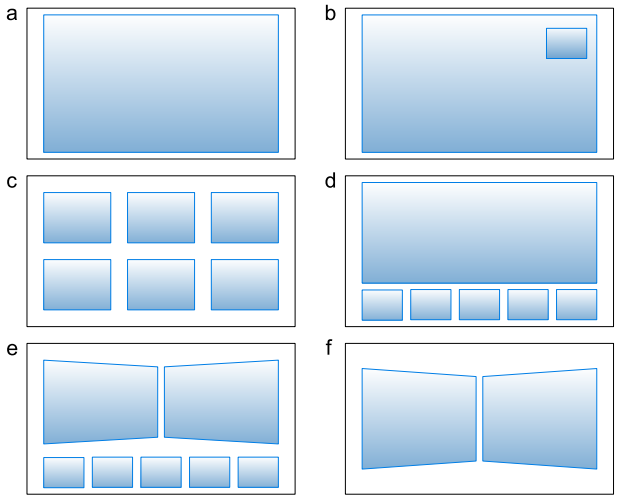

Figure 4: Available layouts: (a) full screen layout; (b) full screen layout with self view in the small region; (c) tiled layout displaying a number of video streams in parallel; (d) layout with focus on one person and with smaller tiles for all other participants (standard layout); (e) split screen layout with two large regions and a number of small previews; (f) split screen layout with two large regions only.

This list described the conditions when to use which layout in a textual form. Table 2 represents these conditions as a set of rules resembling first order logic. It includes the switching conditions as well as further restricting rules, such as preventing switches more frequent than 20 seconds.

3.2.3 *Shot Assignment*

This component fuses the input coming from the layout selection, and the shot identification and prioritisation components. The aim of the shot assignment component is to fill the regions of the selected layout with the ordered list of shots and react to notifications about changes in the conversation. As previously described, each layout has a special purpose and will be used in certain situations. This component knows the semantics of each layout and of the regions inside the layout. Therefore, it is able to assign the right shots into available regions. For example, a shot from a person holding the turn will go into the large region in the center of the layout, as illustrated in Figure 4d, and the previews will be filled up with all others participants.

If the number of participants m exceeds the number of available previews n, then the shot assignment component selects the top n shots based on the ranked list of selected shots. It maintains spatial consistency on the participants' screen by placing the shots always into the same order which is defined by the provided user names. Another principle implemented by this component is to keep awareness of all participants by showing those who were not seen for a long time (see rule 1 in Table 3). Also aesthetic principles are implemented here such as not cutting too fast (see rules 2a and 2b in Table 3).

4. DISCUSSION

The intelligence of the Orchestration Engine is captured by a set of rules divided into individual levels and fulfilling various requirements such as compliance with aesthetic principles (e.g., do not cut too fast) or implement pragmatic principles that focus on communication aspects (e.g., who is currently the active speaker). They are implemented in a declarative manner, using the JBoss Drools[3] reasoning engine. Only the conversation metrics calculator component is implemented imperatively in Java. This knowledge-based approach requires that the intelligence of the Orchestration Engine is defined in advance. Compared to a machine learning approach, the former has the advantage that gives full control over the behaviour and allows to modify, extend, and fine tune the desired behaviour as required. For the current implementation, the intelligence was expressed in English (natural language) and was then refined experimentally, through a process similar to the one described in [8]. This process of defining the intelligence of an Orchestration Engine was preceded by experimental findings in [2] which lead to different approaches and implementations [4, 7, 1]. The concept of orchestration for social groups and how to apply mixing grammar from the domain of TV productions is discussed in [8]. An overview of alternative implementation approaches as well as its pros and cons is discussed in detail in [5, pp. 236–241].

The knowledge elicitation process is already a challenging task on its own [8]. The next challenge is to implement the well defined and complex production grammar as

[3]https://www.jboss.org/drools/

Table 2: Layout selection rules in a notation resembling FOL

1.	$heatedness(low) \wedge monlogue(no) \rightarrow selectLayout(STANDARD)$
2.	$heatedness(high) \wedge monlogue(no) \rightarrow selectLayout(TILED)$
3.	$heatedness(low) \wedge monlogue(yes, P) \wedge isInSameLocation(P) \rightarrow selectLayout(TILED)$
4.	$heatedness(low) \wedge monlogue(yes, P) \wedge isInOtherLocation(P) \rightarrow selectLayout(FULL\_SCREEN)$
5.	$heatedness(low) \wedge dialogue(P1, P2) \rightarrow selectLayout(SPLIT\_SCREEN, P1, P2)$
6.	$heatedness(high) \wedge dialogue(P1, P2) \rightarrow selectLayout(SPLIT\_SCREEN\_WITH\_PREVIEW, P1, P2)$
7a.	$durationOfCurrentLayout < 20sec \rightarrow forceMaintainLayout$
7b.	$durationOfCurrentLayout >= 20sec \rightarrow setFree2SwitchLayout$

Table 3: Shot assignment rules in a notation resembling FOL

1.	$timeSinceLastVisibility(P) > 90sec \wedge isInOtherLocation(P) \rightarrow putInPreview(P)$
2a.	$durationOfCurrentShot < 2sec \rightarrow forceMaintainShot$
2b.	$durationOfCurrentShot > 2sec \rightarrow setFree2Cut$

a set of rules. Several problems come up when the rule-base grows. One of them is of circular dependencies between rules. This is mostly easy to identify and a rules engine also might provide mechanisms in the rules language to avoid such behaviour. Much more difficult to identify and to solve is a circular behaviour when temporal relationships are involved such as "when event A is followed by event B *within 10 seconds* then ...". Also conflicting situations are difficult to solve which always come up when multiple rules make decisions in parallel and when they interfere each other. These issues require careful design and layering of the decision making process into smaller portions as well as explicit conflict resolution mechanisms. The good news is, that evaluations have shown that an Orchestration Engine implemented with JBoss Drools is fast enough for real-time decision making, see also [6]. It does not add a significant delay to the processing chain. At the moment a longitudinal study with a number of experiments within a social network take place to evaluate the Orchestration Engine described in this paper.

5. CONCLUSION

This paper discussed an automatic real-time decision making system, known as Orchestration Engine, in the domain of video mediated communication. We discussed what Orchestration is, what it should consider and how it can work. An Orchestration Engine was implemented consisting of a Semantic Lifter module which fuses and interprets low level events coming from AV analysis into higher level semantically more abstract events. Those events are then used by the Director module which selects a visual layout that fits to the current conversational situation as well as selects the video streams which go in this layout.

The behaviour is implemented declaratively as a set of rules. The knowledge elicitation as well as its implementation is a complex task but this gives us full control over the behaviour and allows us to modify, to extend, and to fine tune the desired behaviour as required.

6. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. ICT-2011-287760. We thank all partners who contributed to the Vconect system for their support to make this work possible.

7. REFERENCES

[1] M. Falelakis, R. Kaiser, W. Weiss, and M. Ursu. Reasoning for video-mediated group communication. In *Proceedings IEEE International Conference on Multimedia & Expo*, pages 1–4, July 2011.

[2] M. Groen, M. Ursu, S. Michalakopoulos, M. Falelakis, and E. Gasparis. Improving video-mediated communication with orchestration. *Comput. Hum. Behav.*, 28(5):1575–1579, Sept. 2012.

[3] F. Hammer, P. Reichl, and A. Raake. The well-tempered conversation: interactivity, delay and perceptual voip quality. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 1, pages 244–249 Vol. 1, May 2005.

[4] R. Kaiser, C. Wagner, M. Hoeffernig, and H. Mayer. The interaction ontology model: supporting the virtual director orchestrating real-time group interaction. In *Proceedings of the 17th international conference on Advances in multimedia modeling*, MMM'11, pages 263–273, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] R. Kaiser and W. Weiss. *Media Production, Delivery and Interaction for Platform Independent Systems: Format-Agnostic Media*, chapter Virtual Director. Wiley, 2014.

[6] R. Kaiser, W. Weiss, M. Falelakis, S. Michalakopoulos, and M. Ursu. A rule-based virtual director enhancing group communication. In *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo Workshops*, pages 187–192, July 2012.

[7] P. Torres and M. Ursu. Induction in first-order logic with temporal metric operators. In *Proceedings of the 21st Conference on Inductive Logic Programming*, 2011.

[8] M. F. Ursu, M. Groen, M. Falelakis, M. Frantzis, V. Zsombori, and R. Kaiser. Orchestration: Tv-like mixing grammars applied to video-communication for social groups. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 333–342, New York, NY, USA, 2013. ACM.