Mirrorverse: Live Tailoring of Video Conferencing Interfaces

Jens Emil Sloth Grønbæk, Marcel Borowski, Eve Hoggan, Wendy E. Mackay, Michel Beaudouin-Lafon, Clemens Nylandsted Klokmose

Mirrorverse: Live Tailoring of Video Conferencing Interfaces

Jens Emil Grønbæk
jensemil@cs.au.dk
Aarhus University
Aarhus, Denmark

Marcel Borowski
marcel.borowski@cs.au.dk
Aarhus University
Aarhus, Denmark

Eve Hoggan
eve.hoggan@cs.au.dk
Aarhus University
Aarhus, Denmark

Wendy E. Mackay
mackay@lisn.fr
Université Paris-Saclay, CNRS, Inria
Orsay, France

Michel Beaudouin-Lafon
mbl@lisn.fr
Université Paris-Saclay, CNRS, Inria
Orsay, France

Clemens N. Klokmose
clemens@cs.au.dk
Aarhus University
Aarhus, Denmark

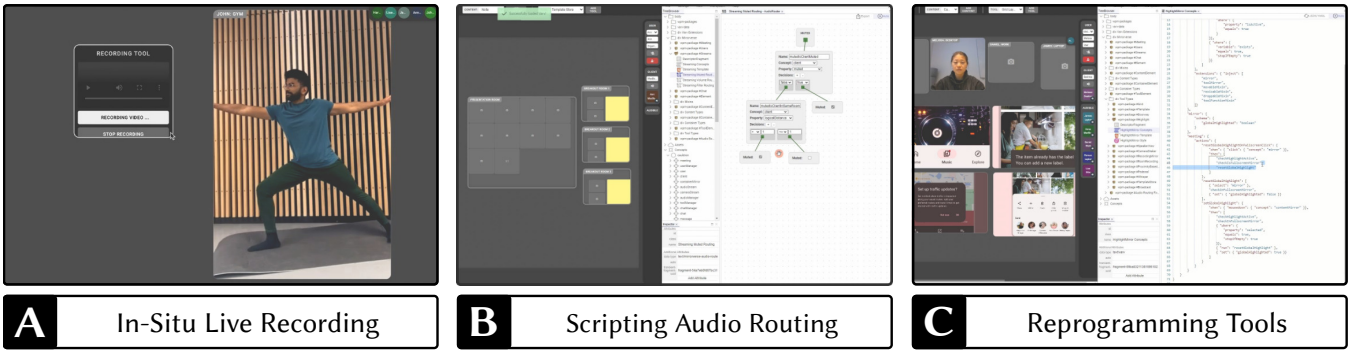

Figure 1: *Mirrorverse* supports live tailoring of video conferencing interfaces. The figure illustrates three novel uses: (A) The recombination of tools enables live recording of camera streams during a yoga instruction session. (B) The audio routing scripting tool enables audio streams of participants to be routed in multi-room workshops. (C) The ability to edit tools enables live reprogramming of their functionality during meetings.

ABSTRACT

How can we let users adapt video-based meetings as easily as they rearrange furniture in a physical meeting room? We describe a design space for video conferencing systems that includes a five-step "ladder of tailorability," from minor adjustments to live reprogramming of the interface. We then present *Mirrorverse* and show how it applies the principles of computational media to support live tailoring of video conferencing interfaces to accommodate highly diverse meeting situations. We present multiple use scenarios, including a virtual workshop, an online yoga class, and a stand-up team meeting to evaluate the approach and demonstrate its potential for new, remote meetings with fluid transitions across activities.

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing systems and tools**; **Collaborative interaction**; *Web-based interaction*; *Interactive systems and tools.*

KEYWORDS

Video conferencing, distributed meetings, end-user tailorability

ACM Reference Format:
Jens Emil Grønbæk, Marcel Borowski, Eve Hoggan, Wendy E. Mackay, Michel Beaudouin-Lafon, and Clemens N. Klokmose. 2023. Mirrorverse: Live Tailoring of Video Conferencing Interfaces. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29-November 1, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3586183.3606767

1 INTRODUCTION

Distributed meetings take many forms depending upon the configuration of people, activities, and shared content. Yet, video conferencing software is traditionally built as one-size-fits-all generic meeting with little support for tailorability of either functionality or the user interface. By contrast, physical meetings let people spontaneously rearrange furniture, equipment, and themselves to match the format and atmosphere of the meeting.

Our goal is to make end-user tailoring of virtual meetings as easy as rearranging physical furniture in a meeting room. Inspired by Mørch [41], we introduce a five-level *ladder of tailorability* — customization, recombination, extension, scripting, and reprogramming — to help us explore what it means to tailor video conferencing software *live while in use.*

UIST '23, October 29-November 1, 2023, San Francisco, CA, USA

We investigate how to reduce *friction* while moving through the levels of this ladder, especially in light of the increasing demand for remote work and the introduction of new, more tailorable apps and services such as Gather's map customization [14] and Zoom's and Teams' app marketplaces [39, 68]. Current systems currently constrain live tailoring capabilities for end-users to include customization, e.g., enabling Zoom's moderation features; and recombination, e.g., combining elements in a room in the Gather map. Although some platforms such as the Zoom Apps Marketplace also support extensions, each requires using a separate development tool chain outside the application and none support automated scripting or new application behavior. Thus our primary research question is: *How can we provide live support for all five tailoring levels in a video conferencing application?*

The paper first analyzes what constitutes video conferencing and explores how applying principles from computational media can enhance tailorability. Next, we present *Mirrorverse*, a proof-of-concept system that shows how live tailoring can be technically realized in a video conferencing interface. We evaluate the approach through demonstration [32] and heuristic analysis [44] using three diverse use scenarios: a virtual workshop activity, an online yoga class, and a stand-up team meeting. These scenarios illustrate how *Mirrorverse*'s live tailoring capabilities enable more dynamic remote meetings with fluid transitions across activities. Tailoring in *Mirrorverse* is admittedly not yet as easy as rearranging furniture. However, it lets users with different levels of technical proficiency live tailor their interface together. This points to a future of more flexible collaborative software that can be adapted to local needs and practices, even while in use.

2 RELATED WORK

We review related research on video-mediated communication and commercial video conferencing tools, and tailorable software.

2.1 Video-Mediated Communication

The earliest video-conferencing systems were developed to permit remote collaboration within a corporation. In the early 1990s, Bly and colleagues introduced the notion of a Media Space [5, 34], which led to a series of novel user interfaces for coping with the fundamental challenges of video-mediated communication. Buxton [9] introduced a taxonomy that divides a video communication space into person, task, and reference space. A consistent reference space is essential for effectively using deixis, i.e. when expressions such as "here" or "there" are accompanied by gestures that point at the shared space. A number of systems have been designed to integrate person and task space which establishes a more seamless reference space [26, 43, 53].

However, few commercial video conferencing tools have taken advantage of this research and have instead relied upon the same small set of standard interfaces since the 1990s [54], namely the "gallery" and "speaker view" layouts. The recent global switch to remote and hybrid work during the COVID-19 pandemic has sparked new interest in video-mediated communication. Recent research has identified key challenges for a *new wave* of video conferencing, including: navigating multiple audio communication channels [4, 23, 24] and backchannel communication [30, 47, 51]; supporting everyday devices not just room hardware [57]; managing turn-taking and the conversational floor [1, 16, 24]; and supporting meeting configuration [18, 46, 49].

This has led to multiple new commercial platforms that use stronger and more flexible spatial metaphors, including Gather [14], SpatialChat [55], Sprout [56], Remo [48], Teamflow [59], Wonder [65] or Ohyay [45]. In parallel, research prototypes have improved our understanding of spatial metaphors [16, 23, 24, 64] and introduced new ways of interacting with live and recorded video streams [18, 25, 60]. Finally, some popular solutions preserve the legacy gallery and speaker views but also integrate task-space apps and add-ons into their workspaces, including Zoom Apps [67, 68], Teams Apps [39], Miro's add-on video-conferencing features [40], and Google's integration of Meet and Docs [17].

Despite some improvements in tailorability, these systems: enforce *low ceilings* — scope is limited to that provided explicitly by the platform; lack *liveliness* — solutions rarely support changes during meetings; and introduce *friction* — transitions to more complex levels of tailoring are difficult. *Mirrorverse* offers a novel video conferencing approach that addresses each of these problems.

2.2 Tailorable Software

Tailorable software, also called *customizable software* [33], *adaptable software* [63], or *personalizable software* [21, 22], refers to software that can be modified and adapted by users to address idiosyncratic needs. Tailoring can occur at different levels [35, 41]. For example, MacLean et al. [35] describe two tailoring dimensions: *tailoring power* — which changes are possible at a particular level and *tailoring skill requirements* — which (technical) skills are needed to tailor the software. For example, changing parameters via a pre-defined menu requires a lower skill level than reprogramming software. Mørch [41] defines three tailoring levels: *customization* — modifying the appearance or presentation of software through predefined configuration options; *integration* — creating a sequence of executions that results in new functionality and *extension* — writing new code to improve or add functionality. Later, we will expand Mørch's [41] *ladder of tailorability* to include five levels.

MacLean et al. [35] further describe how different tailoring techniques should be employed at different levels of tailoring power and skill requirement. Their *Buttons* system lets users tailor functionality on multiple levels, enabling a more "gentle slope" of tailorability compared to the previous example of changing parameters vs. reprogramming. *Mirrorverse* follows a similar approach based on our five-level ladder of tailorability.

For tailoring to occur, users not only need access to technological solutions, but also a culture in which tailoring is the norm [35]. Mackay [33] shows that "customization is not a purely individual activity." Users with different levels of skills can work together, for instance when a peer called a *translator* can help a regular end-user to tailor their software or act as a link between end-users and programmers. These translators are also called *handyman* [35], *gardeners* [13], or *tailors* [62]. More recently, Haraty et al. [22] have found similar roles in online customization sharing.

Dimensions	User Presence			Space			Interaction	
Primitives	**Video**	**Audio**	**Avatar**	**Rooms**	**Transitions**	**Content**	**Views**	**Tools**
Examples								
Zoom, Teams, Discord, Slack, Google Meet	On / Off	Mute / Unmute	Name and image in participant list	Main room and breakout rooms	Discrete movement between rooms	Live streams; Chat	Private layouts (gallery or speaker views)	AV settings; Screen sharing
Gather	Based on proximity	Based on proximity	Game-like avatar and name in participant list	Configured by organizer	Continuous movement; Proximity thresholds	Game map; Avatars; Whiteboards	Private video layout; Shared map view	Spatial interaction with avatars; Add tools ad-hoc
MirrorBlender	Position; Size; Translucency	Mute / Unmute	Video window as embodied avatar	Single main room	None	Camera and screen mirrors	Shared layout, WYSIWIS	Spatial interaction with video
Mirrorverse	Position; Size; Translucency	Scriptable audio routing	All of the above	Rooms treated as elements, Can be nested	Continuous movement; Discrete rooms	Live streams; Chat; Notes; Avatars; …	WYSIWIS workspace; Tool panel; Inspectors; Client panel	Ad-hoc recombination of tools

Table 1: Dimensions and primitives of video conferencing, used to categorize a selection of commercial and research systems.

Mirrorverse supports such practices by enabling translators to tailor the meeting space via programming in the same environment where regular end-users perform customization. This is inspired by a Computational Media approach to software, inspired by Kay's [28] early vision of software as computing clay and diSessa's seminal work [11, 12] on Boxer. These systems blur the line between using and developing software, making tailoring the modus operandi.

Today, the most prominent examples of computational media are spreadsheets and computational notebooks. Although Webstrates [6, 29] demonstrates how to apply the principles of computational media to web-based systems, no current system has applied these principles to video conferencing.

3 DESIGN SPACE FOR TAILORABLE VIDEO CONFERENCING

Based on recent developments in video conferencing platforms and research on end-user tailoring, we articulate a design space for a new wave of tailorable video conferencing.

3.1 Design Space of Video Conferencing

Video conferencing systems let people in different locations communicate in real time through video and audio. Users can see and hear each other, as well as share documents and presentations. Although all these systems support standard gallery and speaker layouts, recent platforms have begun challenging these norms.

Table 1 defines a design space for describing key characteristics of modern video conferencing systems, with examples of each. We identify three main categories: *user presence*, *space*, and *interaction*, and then specify a set of **primitives** (highlighted in bold) that describe common features and differences across these systems.

3.1.1 User Presence. All video conferencing software represents each user's presence and lets them control how they are represented to others. Minimal controls including switching **video** on or off and muting / unmuting the **audio** from their microphone. Unfortunately as meetings become larger, it becomes harder and harder for users to maintain an overview of each others' presence due to limited screen real-estate. To address this issue, most platforms include a meeting **avatar** in the form of an icon, image, figure, or name label in a participant list that continuously represents the user's presence in the meeting.

3.1.2 Space. Most video conferencing systems rely on the **room** as a central metaphor. Newer systems have significantly expanded users' access to the space, e.g., by allowing users to move around a 2D map to support smooth conversational transitions [16]. Systems such as in Gather [14], Sprout [56], MirrorBlender [18], and OpenMic [24] take advantage of the spatial metaphor's notion of proximity to provide access to virtual furniture, breakout rooms or ad-hoc group formations. When we compare the spatial metaphors of Zoom and Gather, we see that their conversational **transitions** can be either *discrete*, e.g., moving instantly between Zoom breakout rooms or *continuous*, e.g., moving within the Gather map.

The spatial metaphor can also divide users from a large virtual meeting into breakout rooms. These rooms provide a scope for a (sub)group and its **content**. Content may include video streams from cameras, screens, or recorded video, images, text notes or chat as well as more complex data such as sketches, or tools, e.g., adding a pedestal in the Gather map to broadcast audio from the speaker.

3.1.3 Interaction. While all platforms support sharing of audio and video streams, they differ in how the interaction is designed, especially how **views** display streams back to the user and which **tools** are available to users. Today's most prominent platforms, including Zoom, Google Meet and Microsoft Teams, offer standard gallery and speaker layouts and standard tools such as screen sharing, virtual backgrounds, chat, recording and presence controls: mute / unmute audio, hide / show video. Some offer special user roles, where, for example, the host may have special permission to control other participants' access and presence.

The new wave of platforms has brought new tools and interaction techniques that take advantage of spatial metaphors, e.g., proximity-based group formations [16] and live shared configuration of video and content [18, 24, 56]. These systems let users adapt the spatial layout to accommodate different collaboration needs and styles.

Levels	Customization	Recombination	Extension	Scripting	Reprogramming
Examples					
Zoom, Teams, Discord, Slack, Google Meet	Pre-customize UI†		App integrations†	Zoom API*	
Gather	Pre-customize the map‡	Combining existing map layouts‡		Gather API*	
MirrorBlender	Ad hoc customizing meeting space‡				Reprogramming in the Cauldron editor*
Mirrorverse	Customize tools with the inspector‡	Adding tools to a room‡; Templates‡	Content and tools are bundled as packages‡	Audio Routing‡	Reprogramming in the Cauldron editor in Varv‡

Table 2: Levels of tailorability with examples of their realization in video conferencing systems (∗ offline; † semi-live; ‡ live).

The trend towards integrating person-space and task-space tools has also increased the need for tailoring, as demonstrated by the way tools have been customized to mitigate "Zoom-bombing" [46] or the main platforms' support for integrating apps [17, 39, 68].

3.2 Tailorability for Video Conferencing

We address three key challenges for integrating tailorability into video conferencing: raising the ceiling, supporting liveliness, and reducing friction.

3.2.1 Raising the Ceiling. We build on Mørch [41] to define five levels of software tailorability — customization, recombination, extension, scripting, and reprogramming — which we map onto the *ladder of tailorability* (see Figure 2).[1] Table 2 maps these levels onto current systems.

Customization adapts software functionality or aesthetics using only predefined settings. For example, users can change Zoom's configuration settings to enable or disable the waiting room or give participants permission to unmute themselves.

Recombination assembles ready-made blocks or the primitives in subsection 3.1 to produce new functionality. For example, Gather lets hosts build rooms and define zones with different functionality.

Extension adds new functionality from external sources. For example, Zoom users can add a Miro board or whiteboard.

Scripting adds new behavior using an exposed API or a built-in scripting language. For example, Zoom offers an exposed API for creating extensions.

Reprogramming changes an application's functionality by editing its source code or by interfacing with it directly. For example, technically, Jitsi [27] supports changing the source code, although this requires rebuilding the entire application before the change takes effect. Smalltalk [15] offers a canonical example of a system that supports reprogramming,

3.2.2 Supporting Liveliness. Video conferencing systems with multiple connected users should be able to support *live* tailoring at each level of the ladder of tailorability during the course of the meeting.

We distinguish among three levels of liveliness when tailoring video conferencing: offline, semi-live, and live: *Offline* requires restarting the meeting to apply a change. *Semi-live* means that changes can be applied while the meeting is running, but the tailoring itself cannot be done collaboratively. For example, an organizer can change the layout of a room in Gather while the meeting is running, but the changes are not applied until they save the new layout. *Live* means that changes can be applied while the meeting is running, *and* tailoring can be done collaboratively. For example, several users could change the configuration of the breakout rooms together, or edit scripts in a shared code editor.

3.2.3 Reducing Friction. In theory, any user should be able to move up the ladder of tailorability from customization to reprogramming, if supported by the system. However, use of video conferencing is socio-technical in nature, and *friction* arises when a user's lack of programming skills, fear of breaking something, or the inherent complexity of the configuration prevents these transitions. Configuration work is a type of articulation work [52]: It is the work to make work work, i.e., in our case, to make video conference meetings work. Configuration causes technical friction, for example, when additional software such as an IDE is required to create an app for Zoom. Users' skills can also cause friction: scripting requires users to write code which requires programming knowledge. However, even when a user has the requisite skills and tools at hand, the potentially large amount of time required for scripting or reprogramming can render the transition insurmountable. We argue that friction can be greatly reduced by simplifying configuration work.

4 MIRRORVERSE

Mirrorverse is a system design and proof-of-concept implementation of video conferencing as computational media (see Figure 3). It was inspired by the metaphor of malleable mirrors in MirrorBlender [18], i.e. live streams that mirror people and screen content and that can be freely arranged in the same virtual workspace.

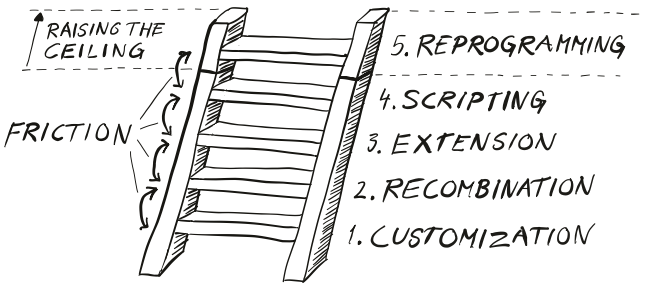

Figure 2: The Ladder of Tailorability: We raise the ceiling of tailorability and reduce the friction of climbing the ladder.

[1] Given changes in software and vocabulary since the 1990s, we have nuanced and expanded on Mørch's original three levels: customization, integration, and extension.

Mirrorverse supports a broad range of meeting experiences by implementing the key primitives of modern video conferencing (described in subsection 3.1) and by supporting live tailorability. For example, *Mirrorverse* can recreate MirrorBlender as well as meeting experiences similar to Zoom, Teams, or Gather. The core design goal is to support meetings that change over time by allowing idiosyncratic adaptation to the needs that arise as a meeting progresses. *Mirrorverse* can be tailored at all five levels of the ladder, and changes are applied *live*, without the need to restart the system.

4.1 Video Conferencing as Computational Media

Mirrorverse applies principles from computational media [6, 29] to realize live tailorability: shareability, malleability, computability, and distributability. *Shareability* means that all aspects of interaction with the software should be possible collaboratively, including changing the software itself. *Malleability* means that it should be possible to change the software to suit individual needs. *Computability* means that it should be possible to treat what is seen on the screen as data for computation or — as diSessa puts it — that the computational structure should be in the accessible parts of the medium [11]. *Distributability* means that software should be able to flexibly span across available devices.

For video conferencing software, these principles mean that we should treat software as a malleable medium within which to conduct meetings, a medium that can be shaped to fit the specific meeting type and meeting culture. This should be possible before, during, and after the meeting and it should be possible to do so collaboratively. It implies that the fundamental primitives and mechanisms of video conferencing must be reified [3] into the user-accessible parts of the medium. For example, so that *where* a participant is located in a meeting space can be used computationally to control their audio (muffled, muted, volume, etc.) in a way that is computationally accessible to the user. Tools and computations should be polymorphic so they can be applied and reused wherever it makes sense to the user [2]. E.g., if a tool can manipulate the pixels of a video feed, it should be applicable to both live and recorded video.

Mirrorverse builds on the Webstrates family of software [6, 29] and the Varv [8] programming model to realize the principles of computational media (details in section 6). *Mirrorverse* is accessed through a Web browser.

4.2 Mirrorverse Overview

The design of *Mirrorverse* is based on a 2D nested canvas,[2] which we call the *workspace*, that combines person space and task space in the same environment — similar to, e.g., Sprout [56]. The space is structured into *rooms* that can contain *elements* that are either *content*, *tools* or nested rooms. Rooms can be navigated and elements can be moved from room to room depending on the use case. Tools make it possible to add functionality to a room — either before or during a meeting. Unlike systems such as Zoom, the layout of a meeting is persisted after a meeting ends so that a meeting can be continued at later points in time or a layout can be prepared beforehand, e.g., by setting up breakout rooms. This layout can be collaboratively manipulated live. Other key concepts include an audio routing mechanism that defines which clients are audible and how, and using recordings during a meeting.

4.2.1 Elements: Rooms, Content, and Tools. *Elements* are either rooms, content, or tools. They are displayed in the workspace as rectangles, which can be selected by clicking them and have options that can be customized with the inspector (see below). New elements can be added using the menu bar by selecting a content type or a tool to be added (see Figure 3A). Existing elements can be removed using the inspector. A room acts as a container for other elements, which can be nested. Our prototype features eight types of content (Camera, Screen, Video, Image, Note, Sketch, Chat, Avatar) and twelve tools (Grid, Speaker View, Layout Template, Doorway, Broadcast, Whisper, Proximity-based Audio, Pedestal, Recording, Room Recording, Template Store, Highlight, Camera Shaker).[3] Both content and tools are located in the same workspace.

Tools are a type of element that adds new functionality to a room, such as a tool for highlighting elements when clicking on them or a tool that arranges elements in a grid layout. A tool's functionality is tied to the room it is located in — tools from other rooms do not affect the current room. Elements and their location and size in the room and workspace follow the What-You-See-Is-What-I-See (WYSIWIS) principle [58]. Content in elements follows a relaxed WYSIWIS model: while the state is the same, the exact visual representation might differ, e.g., the content of a note is shared while the scroll position is private to each client. Similarly, tools might differ in their visual representation and whether they are active, e.g., when a tool is active for one user role but deactivated for another.

4.2.2 Workspace and Tool Panel. The *workspace* is a central fixed-sized 2D canvas (see Figure 3B). It always displays the elements of the current *room* a client is located in. As mentioned above, the canvas is WYSIWIS, so that its elements always have the same location, size and transparency for all clients. Users can move elements around in the canvas, resize them, change their opacity and layering order, and move them in and out of nested rooms.

To the left of the workspace is the *tool panel* (see Figure 3C): When adding a new tool to a room, the tool is by default added to the tool panel. This reduces the clutter in the workspace. However, as an option, a toggle can be switched so that the tool will move to the workspace, allowing users to treat it as a moveable element.

4.2.3 Inspectors. To the right of the workspace are two *inspectors* (see Figure 3D): one for the current room and the other for the currently selected element. They are used to configure their options, e.g., the name of the room or the URL of an image, and to delete elements. For some tools, e.g., the inspector can set the roles for which the tool should be active.

4.2.4 Users, Clients, and the Client Panel. Each device, browser, or browser tab visiting a meeting is a named *client*. Each client, in turn, is connected to a *user*, who has a name and a role. The role can, e.g., be used to restrict a tool's functionality to groups of users. The *client panel* (see Figure 3E) is a sidebar that displays information

[2] While we present *one* design of such a system, the dimensions in our design space could also be used to create other system designs. E.g., we based our design on a 2D user interface, leaving 3D interfaces like virtual and augmented reality to future work.

[3] A full list of element types and their descriptions is available in Appendix A.

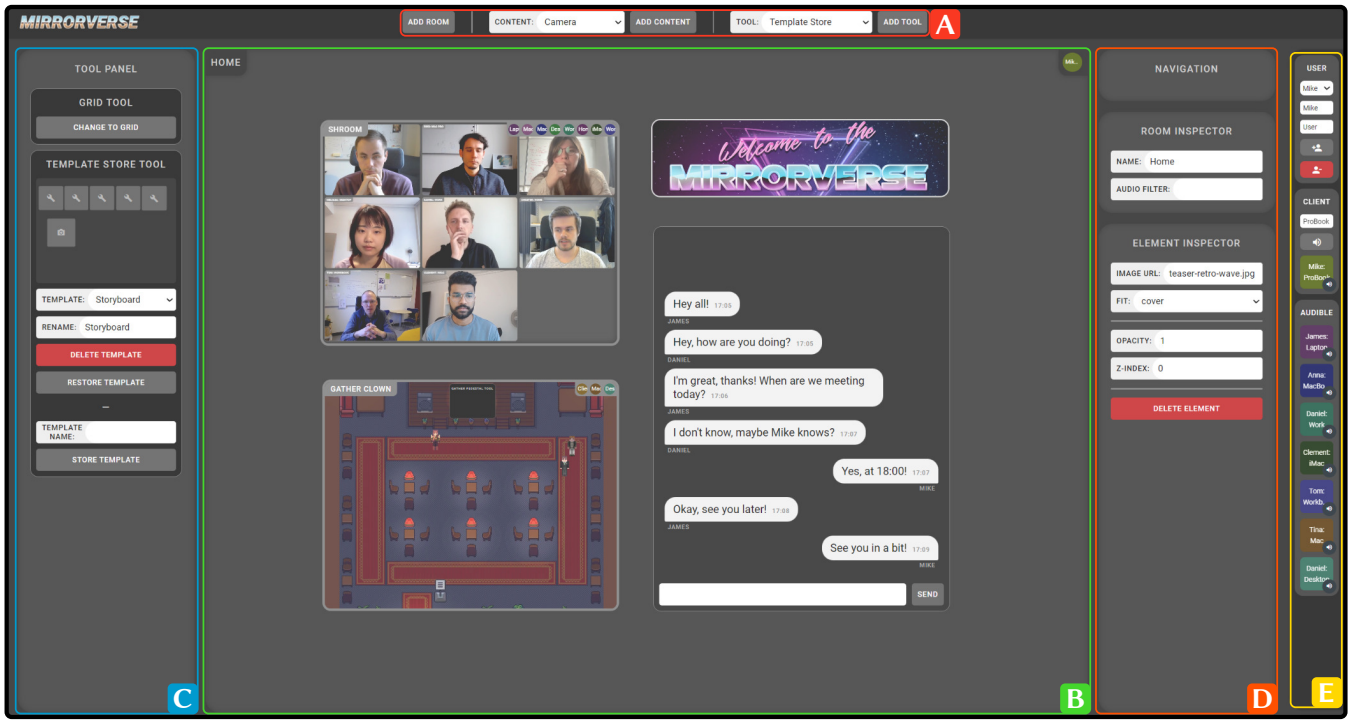

Figure 3: The *Mirrorverse* prototype. New elements can be added using the menu bar (A). Content and room elements are added to the Workspace (B). The top left corner of the Workspace shows the name of the room and breadcrumbs if it is located in other rooms. The top right corner of the Workspace shows all clients currently in the room. Tools can exist as elements in the Workspace but are initially added to the Tool Panel (C). Selecting any element shows its options in the element inspector, e.g., the URL of an image (D). Navigation to move between rooms and options of the current room (room inspector) are placed above that inspector (D). The Client Panel (E) shows the current user, client, and other audible clients in the meeting.

about one's client and user, provides an overview of the clients that are currently audible. Clients that are currently in the same room as oneself are displayed in the top right corner of the workspace. The distinction between clients in the room and clients that are audible is needed since clients from other rooms may be audible (see below in the audio routing section).

4.3 Techniques for Tailorability

Mirrorverse supports tailorability through the following techniques, corresponding to the five levels of the ladder of tailorability (see Figure 2 and Table 2):

4.3.1 Level 1: Customizing Element Options. Elements in *Mirrorverse* have various options that can be customized using inspectors. These may range from preferences, e.g., the color of a note, to more functional options, e.g., which element is considered the "speaker" in a room when using the Speaker View tool. This level of tailorability is typically immediately accessible to users at all levels of proficiency.

4.3.2 Level 2: Recombining Tools and Templates per Room. This level of tailorability allows more technically proficient users, such as meeting hosts, to customize the experience. Tools in *Mirrorverse* are located in rooms and their functionality is limited to the room they are located in. This enables the recombination of different tools (or content) in a room to create different meeting experiences, from existing setups similar to those of Zoom or Gather to novel examples as we demonstrate later.

An example tool is the Recording tool. The ability to record the live media streams of a meeting is common in video conferencing but the recordings can rarely be used *within* the meeting (with a few exceptions, e.g., [25, 60]). *Mirrorverse* supports the recombination of live and recorded videos, so that users can for example demonstrate something to the camera and then play it back and refer to the recording in the meeting (subsection 5.1 demonstrates this).

This and other types of recombination require configuration work, such as setting up rooms with different layouts and different types of tools. The Template Store tool makes it possible to store combinations of tools, content, and layout as templates and restore them later. It effectively reifies such combinations into new objects that can be reused. For example, we created templates that replicate Zoom-like and Gather-like interfaces[4] (see Figure 3) by combining different content elements and tools.

4.3.3 Level 3: Extending Content and Tool Types with Packages. This level of tailorability allows the technically proficient user to seek out new functionality to support a given meeting situation, and for the user who is proficient in programming to share new functionality with other users.

[4]These replicated examples are demonstrated in the accompanying video.

Content types in *Mirrorverse*, such as images, notes or sketches, are bundled as Webstrates packages [7]. Packages can be added and removed from a meeting at run-time, supporting tailoring by extension. If a user implements a new type of content, e.g., a chess game, they can add it to another meeting simply by dragging and dropping it into the Cauldron IDE of the other meeting (see section 6). Adding a tool package works in the same way and extends the list of available tools in the drop-down menu in the toolbar.

4.3.4 Level 4: Scripting the Audio Routing. In *Mirrorverse*, every client has an audio stream with several properties: *muted*, which defines whether the stream is audible; *volume*, which defines the volume of the stream; and *filter*, which defines what audio filter (e.g., a low-pass or high-pass filter) is used on the stream.

The audio routing in *Mirrorverse* is a scripting technique that enables control of the above three properties based on the state of the system. It serves as one example technique at this level of tailorability, which lets users with a high degree of technical proficiency add or modify behavior of the interface without using conventional programming.

The audio routing can be authored either directly in Varv code or by using the audio routing GUI. Each property is calculated using a decision tree that can refer to a variety of states for determining its value. The GUI lets users create one *root node* for each property, *decision nodes* that depend on properties, and *value nodes* that set the value of the property at the leaves of a tree (see Figure 4; subsection 5.2 demonstrates the GUI). Nodes can be created and connected by dragging from the green connection boxes to other nodes. For example, the numeric property `logicalDistance` describes how many rooms a client is away from another client. This property can be used to determine the volume of an audio stream, e.g., if the logical distance of the client is zero, the volume is set to 100 %, if the distance is one it is set to 50 %, and if it is larger than one it is set to 0 % (see Figure 4).

4.3.5 Level 5: Reprogramming Using the Built-in Editor. Everything in *Mirrorverse*, from content types and tools to the core of the system, can be reprogrammed live. This level of tailorability requires programming expertise, but as we will demonstrate later, live collaborative tailoring allows a user without the necessary expertise

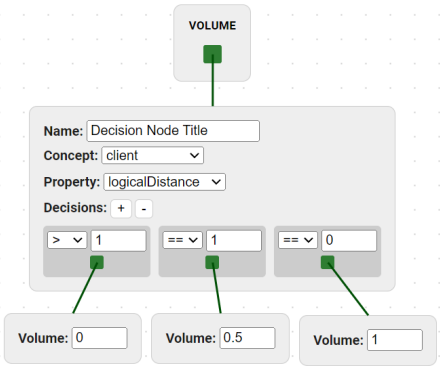

Figure 4: The audio routing GUI. In this example, the volume of audio streams is dependent on the client's logical distance.

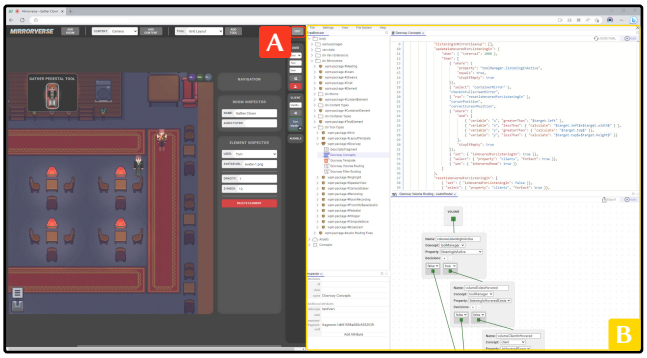

Figure 5: Reprogramming *Mirrorverse* is done using Cauldron, which can be opened using the "Edit" button (A). Here, Cauldron (B) is docked on the right side and shows one editor with Varv code and another with an audio routing GUI.

to realize a change to the interface through the help of a more technically proficient user.

To start reprogramming, users click the "Edit" button in the top right corner (see Figure 5A). This opens the built-in Cauldron IDE, giving access to the code (see Figure 5B). Reprogramming the system happens from within the web browser and no additional tools are required (subsection 5.3 demonstrates this). Edits to Varv concept definitions are immediately applied to the running system live, however, a concept definition can also be disabled while editing.

5 DEMONSTRATING SCOPE AND DEPTH

In the following three usage scenarios, we demonstrate how *Mirrorverse* addresses key challenges of video conferencing: managing audio routing, communicating through video, and adapting tools to impromptu activities. Together, these scenarios demonstrate the breadth and depth of tailoring with *Mirrorverse* through the power of the reification principle.[5]

5.1 Dynamically Recombining Live and Recorded Video in a Yoga Class

Recombining video streams is traditionally done prior to a meeting, e.g., using OBS [42], or after a meeting, e.g., recording and sharing meeting videos. We demonstrate how reified video in *Mirrorverse* enables users to dynamically recombine live and recorded video for remote instruction in an online yoga class (see Figure 6).

5.1.1 Collaboratively Preparing Yoga Sequences. Two yoga instructors, John and Adam, meet to prepare for their next remote yoga class. They first co-create a sequence of yoga poses by taking turns performing and recording different poses (see Figure 6A). Having the videos and the tools in the same workspace enables them to arrange the recording tools in a storyboard layout and brainstorm the sequence while taking turns recording the individual poses.

5.1.2 Arranging Views of the Physical Yoga Studio. John goes to his real yoga studio where he runs the online class. He sets up the camera equipment to point to his yoga mat. He copies the yoga workspace that he made with Adam and "digitally refurnishes it" for his upcoming class: Next to the area of the participants, he creates

[5] All three scenarios are demonstrated in the accompanying video.

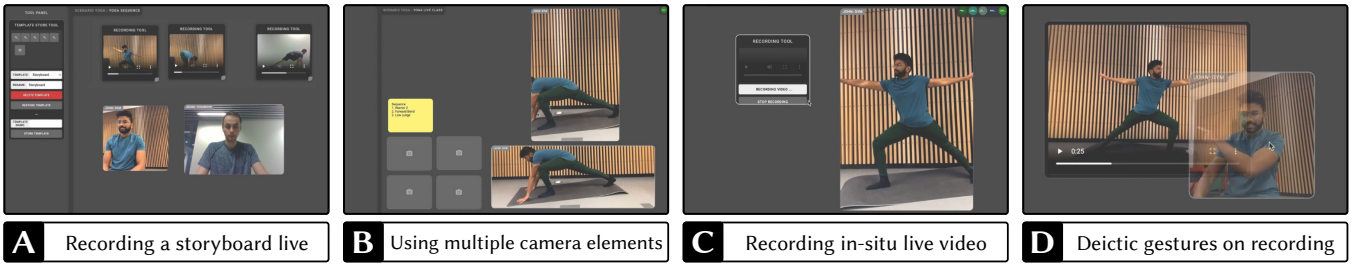

Figure 6: Scenario 1: Dynamically Recombining Live and Recorded Video in a Yoga Class.

two instances of the same camera stream with different crops. One crop shows the mat for portrait-mode shots of poses, the other for landscape mode (see Figure 6B).

5.1.3 Impromptu Recording for Live Demonstration. The yoga session starts. John spots that several participants are doing one of the poses incorrectly with the risk of causing a knee injury. He grabs everyone's attention by dragging his video to the recording tool to record a video of himself performing the pose from the prepared sequence (see Figure 6C), but this time from a different viewing angle that better conveys the potential problem. He then navigates the video clip he just recorded and stops at a specific point to highlight an important detail in the body posture that helps avoid injuries (see Figure 6D).

5.2 Scripting Audio Navigation in a Multi-Room Workshop

Virtual workshop organizers often struggle to navigate audio in multi-room breakout sessions. We demonstrate how *Mirrorverse* reifies audio as the Audio Routing GUI to enable new forms of audio navigation among multiple breakout rooms (see Figure 7).

5.2.1 Preparing the Workshop Interface. To prepare for the upcoming workshop, Ann, the organizer, sets up the meeting with a main room and a set of breakout rooms nested within it (see Figure 7B–C). She anticipates that hearing muffled sounds from the breakout rooms will help her monitor the activity. She opens the Audio Routing GUI (see Figure 7A) and tailors the audio routing so that people in rooms at a logical distance of 1 (i.e. the distance from the main room to the breakout rooms) are still audible but with a low-pass filter applied to convey the notion of distance.

5.2.2 Adding a New Tool for Listening In. With the sound being muffled, Ann cannot hear the contents of the individual conversations, and she finds it cumbersome and disruptive to have to explicitly jump in and out of a breakout room to hear the sound clearly. Instead, she wants an interaction style similar to standing in the doorway of a physical breakout room. She finds and adds the Doorway tool, which allows her to listen in on the audio from a breakout room by hovering over it with the mouse cursor. The tool uses audio routing to lower the logical distance to 0 for the room at the cursor position (see Figure 7D). Finally, Ann wraps up the breakout activity by using the Broadcast tool, which sends her audio to all rooms, to ask participants to go back to the main room.

5.3 Impromptu Reprogramming of Deictic Tools in a Stand-Up Meeting

Distributed teams often need to tailor their workspace to the specific collaboration needs arising during an impromptu discussion. We demonstrate how the reification of tools that can be reprogrammed live can be used to support transitions from planned to ad hoc activities in a design team's stand-up meeting (see Figure 8).

5.3.1 Adding a New Tool for Highlighting Elements. To prepare for their daily stand-up meeting, Roman, the project manager, restores the "Stand-Up Meeting" template using the Template Store tool (see Figure 8A). It includes the Grid and Highlight tools. The latter can be used to manage turn-taking in their status round. With the tool enabled, Roman can click on a participant's video to highlight it for everyone (see Figure 8B). This can accompany deictic expressions when referring to others as "you," rather than using their names.

5.3.2 Tools are Polymorphic Across Person and Task Space. Daniel, a designer, and Melissa, a programmer, get into an impromptu discussion about an urgent task. Roman decides to pause the status round and facilitate the discussion. They reorganize the workspace to focus on the task (see Figure 8C). As they switch their activity to sharing, producing, and arranging content together in the digital task space, they repurpose the Highlight tool to support deictic expressions related to the shared notes and images. This is possible because tools in *Mirrorverse* are polymorphic, meaning that they can be used on different types of content elements.

5.3.3 Reprogramming the Highlight Tool. The current Highlight tool can only select one element at a time. During the coffee break, Melissa and Roman reprogram the tool to enable the selection of multiple elements (see Figure 8D). Participants can now use deictic expressions such as "these two" or "you three" while referring to groups of people or content.

5.3.4 Storing the New Workspace as a Template. As they wrap up their meeting, they collectively agree that their new workspace layout and tool combination worked well and they want to keep it. Roman stores the final workspace as a template for future reuse.

5.4 Other Examples

To further demonstrate the power of reification and live tailoring in *Mirrorverse*, we implemented a few additional examples.

5.4.1 Audio-based Backchannel Communication. We used the audio routing to implement a Whisper tool that enables backchannel

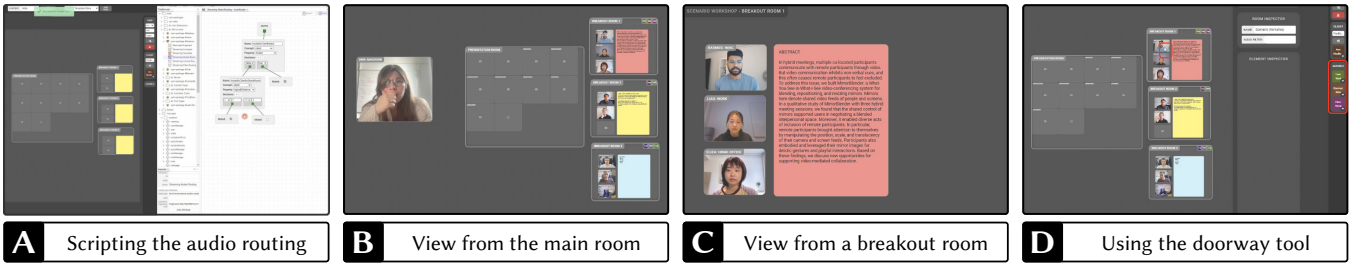

Figure 7: Scenario 2: Scripting Audio Navigation in a Multi-Room Workshop. Notice the cursor hovering over a room in (D), causing the audible clients in the client panel to change to the clients in the hovered room.

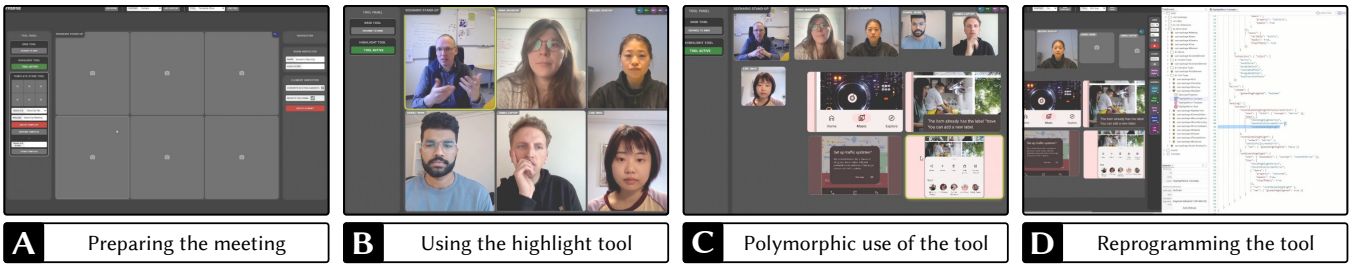

Figure 8: Scenario 3: Impromptu Reprogramming of Deictic Tools in a Stand-Up Meeting.

communication. It works like the Doorway tool, but instead of hovering over a room to listen in, users can hover over a participant's video with their cursor to have a short one-to-one conversation with that participant, which others cannot hear.

5.4.2 Audio as Input to Modify Video Output. To demonstrate the flexibility of audio reification, we implemented a Camera Shaker tool that uses audio as input to modify the video's appearance. It can be used to amplify and enhance non-verbal cues, such as making the video window shake when a person is too loud, i.e., when the volume exceeds a user-defined threshold.

5.4.3 Distributing Content Elements Across Devices. Using the principle of transclusion [29] from the underlying Webstrates platform, it is possible to distribute content elements across multiple screens and devices. We implemented a self-contained *chat mechanism* that can be both embedded in a meeting as content and accessed directly on a companion device (e.g., a mobile phone; see Figure 9).

6 IMPLEMENTATION

Mirrorverse[6] is built on top of the Webstrates software stack consisting of Webstrates [29], Codestrates v2 [7] and Varv [8]. It is implemented in Varv, HTML, and CSS. Extensions to Varv are written in JavaScript. It runs client-side in a Web browser and can be modified using Codestrates v2's built-in IDE, Cauldron.

6.1 Web-based Software Stack

Webstrates serves web pages, called *webstrates*, so that the document object model (DOM), including embedded JavaScript and CSS, is synchronized in real-time between clients and persisted on the server. A webstrate is self-contained and identified by its URL. Webstrates, further, allows for WebRTC streaming of audio and video channels. Codestrates v2 [7] is a development platform for Webstrates consisting of an execution engine and a package manager. Users author code via the Cauldron IDE from within a webstrate, without additional tools. *Mirrorverse* uses the state synchronization and WebRTC streaming capabilities of Webstrates, and the package manager and collaborative code editor of Codestrates.

Varv [8] is built on top of this stack as a new programming model for computational media. It supports authoring interactive behavior of software live and collaboratively. It uses a declarative data structure written in JSON to define interaction.[7] The declarative model enables *accretive extensibility*: applications in Varv are inherently extensible and their interactive behavior can be changed by *adding* code instead of having to *modify* existing code — similar to how CSS rules can be overwritten. A view layer supports creating GUIs using HTML templates and CSS. Varv persists the program's state in the DOM, which is synchronized among clients by Webstrates.

Using the Webstrates stack and Varv was a pragmatic choice as they support collaboration and live reprogramming within the same medium. Webstrates could be substituted with other client-server architectures such as a Node.js server, and Codestrates by other web-based code editors such as CodeMirror [10] and a package manager like NPM. Varv, however, was introduced as a novel programming model and we are not aware of other models with similar capabilities.

In terms of the ladder of tailorability, techniques at the levels of customization, recombination, and extension are mostly independent of the stack. It is only the techniques at the levels of scripting and reprogramming that rely on Varv. Yet, these techniques would also be applicable to other future live programming models.

[6] Mirrorverse on GitHub: https://github.com/Webstrates/Mirrorverse

[7] A code example of Varv is provided in Appendix B.

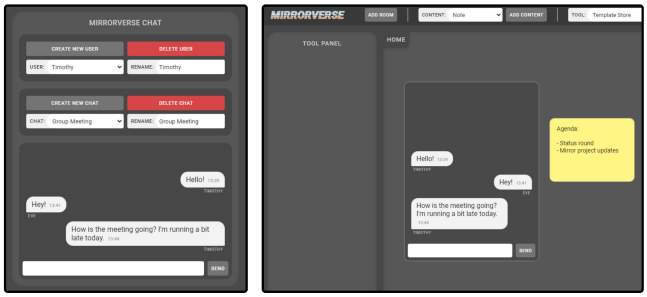

Figure 9: A dedicated chat application (left) can run on a different device but connect to the same chat messages as Mirrorverse (right) using Varv's data store mechanism.

6.2 Architecture

The code of *Mirrorverse* is structured in two types of packages. The first type are low-level extensions to Varv implemented in JavaScript. Among other things, they provide interfaces to the WebRTC [38] and Web Audio [37] APIs, and enable recording video. The second type are packages that implement the *Mirrorverse* platform. Each of these packages contains a combination of Varv code, HTML templates, CSS styling, and audio routings. They are structured into *managers* for handling users, audio and video streams, tools, and the chat, and *elements* that implement rooms, content, and tools. The managers and the room elements are the core features of *Mirrorverse*, all other content and tool element types (e.g., Camera, Image, Grid tool) are implemented as independent packages. This enables adding and removing packages at runtime because dependencies between elements are rare.[8] Currently, *Mirrorverse* supports eight content types and twelve tools (see Appendix A). However, as *Mirrorverse* builds on the power of web technologies, new content types can, e.g., be created by reusing existing JavaScript libraries or by embedding existing web apps.

6.3 The Audio Routing Mechanism

Audio routing is implemented using both Varv and JavaScript. It is part of the *Mirrorverse* infrastructure and can be used by itself in Cauldron or when creating tools. The routing decision tree is defined in Varv — this is necessary to allow for the routing to be changed live during a meeting. Once the properties of audio streams are set in the Varv part of the routing, they are saved to the DOM, where they are picked up by the JavaScript part of the audio routing. The JavaScript part reads the properties from the DOM and changes the corresponding properties on the WebRTC stream using the Web Audio API, e.g., using a `GainNode` to change the volume.

The audio routing GUI (see Figure 4) was created on top of the audio routing mechanism as an authoring tool. It is implemented in JavaScript and integrated into the Cauldron editor of the Webstrates stack. Decision trees created with the audio routing GUI are exported as Varv code used by the audio routing mechanism. This also enables an escape hatch by allowing to author an audio routing using the GUI for the overall tree and implementing advanced features directly in the exported Varv code.

[8]A few packages have dependencies, e.g., the recording tool requires camera content.

7 DISCUSSION

We discuss the key implications of our work and evaluate *Mirrorverse* using Olsen's heuristics for user interface systems [44]. (Relevant qualities are highlighted in bold).

7.1 Systems-Oriented Evaluation

Enabling tailoring of video conferencing **empowers participants to become designers**. Lowering the friction in climbing the ladder of tailorability facilitates the transition from user to designer, and encourages participants to design or change a feature in collaboration with a more experienced colleague.

A *Mirrorverse* meeting is a Varv program consisting of a set of primitives specific to video conferencing (see subsection 3.1). Because a Varv program is a data structure, it is also easy to generate Varv code procedurally. This means that a domain-specific scripting tool such as the audio routing GUI can be built without having to explicitly create an API for it (as required by others, see Table 2). Thus, programming with *Mirrorverse* improves **expressive match** as it encourages creating and using domain-specific abstractions [8] i.e. design thinking in terms of video conferencing primitives.

Mirrorverse supports **inductive combination** at multiple levels. For example, tool elements are primitives that can be recombined in countless ways. This lets users add new tools via extension thus enabling multiple new use situations not covered in this paper. Fundamentally, tailorability offers **flexibility** and liveness improves flexibility. Users can rapidly experiment with new features even during an ongoing meeting with multiple participants.

7.2 Tailorability, Friction, and Usability

We have demonstrated how *Mirrorverse* reduces friction from a technical perspective by making tailorability possible at each level and by enabling live transitions between them. However, future work is necessary to evaluate if and how users will develop, use and combine tools with *Mirrorverse*, and whether or not this significantly reduces friction under real-world use.

Mirrorverse poses several usability challenges. While *Mirrorverse*'s ability to shift from configuration to implementation work while running a meeting raises the ceiling of tailorability, it also introduces a usability trade-off, since the increased flexibility also increases the risk of breaking the system as it runs. Borowski et al. [6] argue that computational media needs safeguards to prevent accidental breakage and should always support easy reversion to a previous working version. *Mirrorverse* let's users revert changes both to the source code and/or to the application state (e.g., the precise configuration of a meeting at a given point in time) using the version history of Webstrates [29].

Even without errors, live reconfiguration can be disruptive and distracting. Organizers of physical meetings handle this by preparing for changes in advance. Digital meeting organizers should also be able to prepare changes "on the side," before making them live. For example, an organizer could rearrange breakout rooms during a plenary session, before sharing them with everyone. Moreover, building and recombining tools also require *work*. We provide Template and Extension Stores to facilitate easy reuse and make this work a community effort, which should reduce friction.

Another usability challenge is awareness: when live changes made by a user affect other users, awareness mechanisms should help the other users understand the change. This is a well-known challenge in groupware research, and this literature could inform good mechanisms for enhancing awareness of the *who, what, and where* [20] of live collaborative tailoring.

7.3 Performance and Scalability

Video and audio communication use only peer-to-peer communication using WebRTC. This does not scale well beyond a handful of users. Handling large meetings would require a media server for multiplexing video and audio streams in order to reduce resource demands on clients. *Mirrorverse* performs well in relatively complex room setups. However, there are numerous optimizations to be done in the Varv runtime engine that would increase performance.

Another scalability issue relates to the approach of recombining tools. As the collection of extensions and templates grows, recombination may lead to conflicts that cause unexpected behavior for the users. For example, two different tools added to the same meeting may lead to unexpected results or even errors if they operate on the same data, such as conflicting audio routing logic.

7.4 Hybrid Meetings

To reduce the complexity in our usage scenarios, we have focused on conventional video conferencing situations where participants connect from each their own device. But in hybrid meetings, physically co-located participants may share a device, such as a large meeting display, to connect to one or more remote participants. These types of meetings have become commonplace, but are also known to be fraught with challenges [50].

The core interactions in *Mirrorverse* replicate the experience of MirrorBlender [18], i.e., by letting users reposition, resize, and adjust translucency of their video windows. These interactions were originally designed to address challenges with deictic referencing in hybrid meetings. Although out of scope for this paper, the malleability of *Mirrorverse* provides a great foundation for further experimentation with new ways to address these challenges in hybrid meetings.

7.5 Directions for Future Work

Our design space covers the current trend of commercial video conferencing platforms and recent research prototypes, or what is sometimes referred to as the 2D Metaverse [31]. However, future work should consider support for configuring 3D environments, such as mixed and virtual realities [19, 66] and cross-device video conferencing [36]. We demonstrated how to move chat windows onto a companion device. This type of cross-device interaction could potentially be extended to new interactive devices such as head-mounted displays.

Future work should further explore the possibilities of reifying video conferencing primitives into interactive objects. Our examples of applying filters and rules to video and audio only show the beginning. An obvious next direction is to reify *text* as a key data format. Video conferencing includes a variety of text content, such as chat, embedded notes, and live transcriptions. With the flexibility and support for quickly experimenting with such different parts of the interface, *Mirrorverse* is an ideal platform for conducting studies on how to improve different aspects of online meetings.

Finally, our demonstration of applying principles of computational media hints at a future where video conferencing is not trapped inside an application, but part of the computational fabric itself, allowing for deeper integration of person and task space.

8 CONCLUSION

The ongoing global switch to hybrid work calls for a new breed of video conferencing systems that can be more easily configured by end users to fit their needs. We have introduced a design space for live tailoring of video conferencing interfaces and a system design based on principles of computational media that spans all five levels of the ladder of tailorability. We have described *Mirrorverse*, a proof-of-concept implementation of this design, and have demonstrated through a series of examples the range of video conferencing experiences and reconfiguration capabilities that it enables. *Mirrorverse* extends tailoring in two important ways: (1) it increases the ceiling of tailorability with reprogramming, and (2) it reduces tailoring friction by supporting live collaborative changes and providing constructs, such as templates and a domain-specific language, to simplify configuration work. Future work includes extending this approach to 3D environments and multi-device settings, as well as exploring the reification of more video conferencing primitives.

ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 740548). Thanks to Rolf Bagge and Janus Bager Kristensen from CAVI at Aarhus University for their support in the development of Mirrorverse.

REFERENCES

[1] Nancy Baym, Rachel Bergmann, Adam Coleman, Ricardo Reyna Fernandez, Sean Rintel, Abigail Sellen, and Tiffany Smith. 2021. *Collaboration and Meetings - Chapter 1 of the 2021 New Future of Work report.* Microsoft, Redmond, WA, USA, Chapter 1, 7–17. https://www.microsoft.com/en-us/research/publication/collaboration-and-meetings/

[2] Michel Beaudouin-Lafon. 2000. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. In *Proceedings of the 18th International Conference on Human Factors in Computing Systems (CHI '00).* ACM, New York, NY, USA, 446–453. https://doi.org/10.1145/332040.332473

[3] Michel Beaudouin-Lafon and Wendy E. Mackay. 2000. Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Palermo, Italy) *(AVI '00).* Association for Computing Machinery, New York, NY, USA, 102–109. https://doi.org/10.1145/345513.345267

[4] Anna Bleakley, Daniel Rough, Justin Edwards, Philip Doyle, Odile Dumbleton, Leigh Clark, Sean Rintel, Vincent Wade, and Benjamin R. Cowan. 2022. Bridging social distance during social distancing: exploring social talk and remote collegiality in video conferencing. *Human–Computer Interaction* 37, 5 (2022), 404–432. https://doi.org/10.1080/07370024.2021.1994859

[5] Sara A. Bly, Steve R. Harrison, and Susan Irwin. 1993. Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment. *Commun. ACM* 36, 1 (1993), 28–46. https://doi.org/10.1145/151233.151235

[6] Marcel Borowski, Bjarke V. Fog, Carla F. Griggio, James R. Eagan, and Clemens N. Klokmose. 2022. Between Principle and Pragmatism: Reflections on Prototyping Computational Media with Webstrates. *ACM Trans. Comput.-Hum. Interact.* (oct 2022). https://doi.org/10.1145/3569895 Just Accepted.

[7] Marcel Borowski, Janus Bager Kristensen, Rolf Bagge, and Clemens N. Klokmose. 2021. *Codestrates v2: A Development Platform for Webstrates.* Technical Report. Aarhus University. https://pure.au.dk/portal/en/publications/codestrates-v2-a-development-platform-for-webstrates(66e1d4d9-27da-4f6b-85b3-19b0993caf22).html

[8] Marcel Borowski, Luke Murray, Rolf Bagge, Janus Bager Kristensen, Arvind Satyanarayan, and Clemens Nylandsted Klokmose. 2022. Varv: Reprogrammable Interactive Software as a Declarative Data Structure. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 492, 20 pages. https://doi.org/10.1145/3491102.3502064

[9] Bill Buxton. 2009. *Mediaspace – Meaningspace – Meetingspace*. Springer London, London, 217–231. https://doi.org/10.1007/978-1-84882-483-6_13

[10] CodeMirror. 2023. CodeMirror. https://codemirror.net/. Accessed: 2023-07-04.

[11] Andrea A. diSessa. 2001. *Changing Minds: Computers, Learning, and Literacy*. MIT Press, Cambridge, MA, USA. https://mitpress.mit.edu/books/changing-minds

[12] Andrea A. diSessa and Harold Abelson. 1986. Boxer: A Reconstructible Computational Medium. *Commun. ACM* 29, 9 (1986), 859–868. https://doi.org/10.1145/6592.6595

[13] Michelle Gantt and Bonnie A. Nardi. 1992. Gardeners and Gurus: Patterns of Cooperation among CAD Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 107–117. https://doi.org/10.1145/142750.142767

[14] Gather Presence, Inc. 2023. Gather. https://gather.town/. Accessed: 2023-07-04.

[15] Adele Goldberg and David Robson. 1983. *Smalltalk-80: the language and its implementation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[16] Carlos Gonzalez Diaz, John Tang, Advait Sarkar, and Sean Rintel. 2022. Making Space for Social Time: Supporting Conversational Transitions Before, During, and After Video Meetings. In *2022 Symposium on Human-Computer Interaction for Work* (Durham, NH, USA) *(CHIWORK 2022)*. Association for Computing Machinery, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.1145/3533406.3533417

[17] Google LLC. 2023. Google Workspace. https://workspace.google.com. Accessed: 2023-07-04.

[18] Jens Emil Grønbæk, Banu Saatçi, Carla F. Griggio, and Clemens Nylandsted Klokmose. 2021. MirrorBlender: Supporting Hybrid Meetings with a Malleable Video-Conferencing System. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 451, 13 pages. https://doi.org/10.1145/3411764.3445698

[19] Jens Emil Grønbæk, Ken Pfeuffer, Eduardo Velloso, Morten Astrup, Melanie Sønderkær Pedersen, Martin Kjær, Germán Leiva, and Hans Gellersen. 2023. Partially Blended Realities: Aligning Dissimilar Spaces for Distributed Mixed Reality Meetings. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, 16 pages. https://doi.org/10.1145/3544548.3581515

[20] Carl Gutwin and Saul Greenberg. 2002. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)* 11 (2002), 411–446.

[21] Mona Haraty and Joanna McGrenere. 2016. Designing for Advanced Personalization in Personal Task Management. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. ACM, New York, NY, USA, 239–250. https://doi.org/10.1145/2901790.2901805

[22] Mona Haraty, Joanna McGrenere, and Andrea Bunt. 2017. Online Customization Sharing Ecosystems: Components, Roles, and Motivations. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*. ACM, New York, NY, USA, 2359–2371. https://doi.org/10.1145/2998181.2998289

[23] Erzhen Hu, Md Asshikur Rahman Azim, and Seongkook Heo. 2022. FluidMeet: Enabling Frictionless Transitions Between In-Group, Between-Group, and Private Conversations During Virtual Breakout Meetings. In *Proceedings of the 40rd Annual ACM Conference on Human Factors in Computing Systems* (New Olearns, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, 17 pages. https://doi.org/10.1145/3491102.3517558

[24] Erzhen Hu, Jens Emil Sloth Grønbæk, Austin Houck, and Seongkook Heo. 2023. OpenMic: Utilizing Proxemic Metaphors for Conversational Floor Transitions in Multiparty Video Meetings. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 793, 17 pages. https://doi.org/10.1145/3544548.3581013

[25] Erzhen Hu, Jens Emil Sloth Grønbæk, Wen Ying, Ruofei Du, and Seongkook Heo. 2023. ThingShare: Ad-Hoc Digital Copies of Physical Objects for Sharing Things in Video Meetings. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 365, 22 pages. https://doi.org/10.1145/3544548.3581148

[26] Hiroshi Ishii, Minoru Kobayashi, and Jonathan Grudin. 1993. Integration of Interpersonal Space and Shared Workspace: ClearBoard Design and Experiments. *ACM Trans. Inf. Syst.* 11, 4 (1993), 349–375. https://doi.org/10.1145/159764.159762

[27] Jitsi. 2023. Jitsi Meet. https://meet.jit.si/. Accessed: 2023-07-04.

[28] Alan Kay and Adele Goldberg. 1977. Personal Dynamic Media. *Comput. J.* 10, 3 (1977), 31–41. https://doi.org/10.1109/c-m.1977.217672

[29] Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 280–290. https://doi.org/10.1145/2807442.2807446

[30] Anastasia Kuzminykh and Sean Rintel. 2020. Classification of functional attention in video meetings. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–13.

[31] Jens Larsen. 2022. Virtual communication without sniper rifles: How the 2D metaverse is changing virtual spaces. https://venturebeat.com/games/virtual-communication-without-sniper-rifles-how-the-2d-metaverse-is-changing-virtual-spaces/. Accessed: 2023-07-04.

[32] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/3173574.3173610

[33] Wendy E. Mackay. 1990. Patterns of Sharing Customizable Software. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work (CSCW '90)*. ACM, New York, NY, USA, 209–221. https://doi.org/10.1145/99332.99356

[34] Wendy E Mackay. 1999. Media spaces: Environments for informal multimedia interaction. *Computer Supported Co-operative Work* 7 (1999), 55–82.

[35] Allan MacLean, Kathleen Carter, Lennart Lövstrand, and Thomas Moran. 1990. User-Tailorable Systems: Pressing the Issues with Buttons. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*. ACM, New York, NY, USA, 175–182. https://doi.org/10.1145/97243.97271

[36] Nicolai Marquardt, Nathalie Henry Riche, Christian Holz, Hugo Romat, Michel Pahud, Frederik Brudy, David Ledo, Chunjong Park, Molly Jane Nicholas, Teddy Seyed, Eyal Ofek, Bongshin Lee, William A.S. Buxton, and Ken Hinckley. 2021. AirConstellations: In-Air Device Formations for Cross-Device Interaction via Multiple Spatially-Aware Armatures. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 1252–1268. https://doi.org/10.1145/3472749.3474820

[37] MDN Contributors . 2023. Web Audio API. https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API. Accessed: 2023-07-04.

[38] MDN Contributors. 2023. WebRTC API. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API. Accessed: 2023-07-04.

[39] Microsoft. 2023. Microsoft Teams Marketplace. https://appsource.microsoft.com/en-us/marketplace/apps?product=teams&exp=ubp8. Accessed: 2023-07-04.

[40] Miro. 2023. Miro. https://miro.com/. Accessed: 2023-07-04.

[41] Anders Mørch. 1997. Three levels of end-user tailoring: Customization, integration, and extension. *Computers and design in context* 1997 (1997), 61.

[42] OBS. 2023. OBS: Open Broadcaster Software. https://obsproject.com. Accessed: 2023-07-04.

[43] Kenton O'Hara, Jesper Kjeldskov, and Jeni Paay. 2011. Blended Interaction Spaces for Distributed Team Collaboration. *ACM Trans. Comput.-Hum. Interact* 18, 1, Article 3 (2011), 28 pages. https://doi.org/10.1145/1959022.1959025

[44] Dan R. Olsen. 2007. Evaluating User Interface Systems Research. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology* (Newport, Rhode Island, USA) *(UIST '07)*. Association for Computing Machinery, New York, NY, USA, 251–258. https://doi.org/10.1145/1294211.1294256

[45] Pine Labs LLC. 2023. ohyay. https://ohyay.co/. Accessed: 2023-07-04.

[46] Washington Post. 2023. Remember Zoom-bombing? This is how Zoom tamed meeting intrusions. https://www.washingtonpost.com/technology/2023/01/24/zoom-bombing-prevention-tips/. Accessed: 2023-03-06.

[47] Irene Rae, Gina Venolia, John C. Tang, and David Molnar. 2015. A Framework for Understanding and Designing Telepresence. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. Association for Computing Machinery, New York, NY, USA, 1552–1566. https://doi.org/10.1145/2675133.2675141

[48] Remo. 2023. Remo. https://remo.co/. Accessed: 2023-07-04.

[49] Banu Saatçi, Kaya Akyüz, Sean Rintel, and Clemens Nylandsted Klokmose. 2020. (Re)Configuring Hybrid Meetings: Moving from User-Centered Design to Meeting-Centered Design. *Computer Supported Cooperative Work (CSCW)* 29 (2020), 769–794. https://doi.org/10.1007/s10606-020-09385-x

[50] Banu Saatçi, Roman Rädle, Sean Rintel, Kenton O'Hara, and Clemens Nylandsted Klokmose. 2019. Hybrid meetings in the modern workplace: stories of success and failure. In *Collaboration Technologies and Social Computing: 25th International Conference, CRIWG+ CollabTech 2019, Kyoto, Japan, September 4–6, 2019, Proceedings 25*. Springer, Springer, Hanover, PA, USA, 45–61.

[51] Advait Sarkar, Sean Rintel, Damian Borowiec, Rachel Bergmann, Sharon Gillett, Danielle Bragg, Nancy Baym, and Abigail Sellen. 2021. The Promise and Peril of Parallel Chat in Video Meetings for Work. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI EA '21)*. Association for Computing Machinery, New York, NY, USA, Article 260, 8 pages. https://doi.org/10.1145/3411763.3451793

[52] Kjeld Schmidt and Liam Bannon. 1992. Taking CSCW seriously: Supporting articulation work. *Computer Supported Cooperative Work (CSCW)* 1 (1992), 7–40.
[53] Abigail Sellen, Bill Buxton, and John Arnott. 1992. Using Spatial Cues to Improve Videoconferencing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Monterey, California, USA) *(CHI '92)*. Association for Computing Machinery, New York, NY, USA, 651–652. https://doi.org/10.1145/142750.143070
[54] Abigail J. Sellen. 1995. Remote Conversations: the Effects of Mediating Talk With Technology. *Human-Computer Interaction* 10, 4 (1995), 401–444. https://doi.org/10.1207/s15327051hci100_2
[55] SpatialChat. 2023. SpatialChat. https://spatial.chat/. Accessed: 2023-07-04.
[56] Sprout. 2023. Sprout. https://sprout.place/. Accessed: 2023-07-04.
[57] Steelcase Inc. 2023. Steelcase: Hybrid Collaboration is Hard. https://www.steelcase.com/eu-en/research/articles/topics/hybrid-work/hybrid-collaboration-hard/. Accessed: 2023-07-04.
[58] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar. 1987. WYSIWIS Revised: Early Experiences with Multiuser Interfaces. *ACM Trans. Inf. Syst.* 5, 2 (1987), 147–167. https://doi.org/10.1145/27636.28056
[59] Teamflow. 2023. Teamflow. https://www.teamflowhq.com/. Accessed: 2023-07-04.
[60] Balasaravanan Thoravi Kumaravel, Fraser Anderson, George Fitzmaurice, Bjoern Hartmann, and Tovi Grossman. 2019. Loki: Facilitating Remote Instruction of Physical Tasks Using Bi-Directional Mixed-Reality Telepresence. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 161–174. https://doi.org/10.1145/3332165.3347872
[61] tldraw. 2023. tldraw. https://www.tldraw.com/. Accessed: 2023-07-04.
[62] Randall H. Trigg and Susanne Bødker. 1994. From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)*. ACM, New York, NY, USA, 45–54. https://doi.org/10.1145/192844.192869
[63] Randall H. Trigg, Thomas P. Moran, and Frank G. Halasz. 1987. Adaptability and Tailorability in NoteCards. In *Human-Computer Interaction—INTERACT '87*. North-Holland, Amsterdam, Netherlands, 723–728. https://doi.org/10.1016/B978-0-444-70304-0.50117-5
[64] Unhangout. 2023. Unhangout. https://unhangout.media.mit.edu/. Accessed: 2023-07-04.
[65] Wonder. 2021. Wonder. https://www.wonder.me. Accessed: 2021-12-08.
[66] Haijun Xia, Sebastian Herscher, Ken Perlin, and Daniel Wigdor. 2018. Spacetime: Enabling Fluid Individual and Collaborative Editing in Virtual Reality. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 853–866. https://doi.org/10.1145/3242587.3242597
[67] Zoom Video Communications, Inc. 2023. Zoom. https://www.zoom.us. Accessed: 2023-07-04.
[68] Zoom Video Communications, Inc. 2023. Zoom Marketplace. https://marketplace.zoom.us. Accessed: 2023-07-04.

A MIRRORVERSE ELEMENT TYPES

A.1 Content Types

Name	Description
Camera	Displays a live camera stream.
Screen	Displays a live screen capture stream.
Video	Plays a video file.
Image	Displays an image file.
Note	Displays a plain text note similar to a post-it.
Sketch	Displays a 2D sketch drawing board using an transcluded tldraw [61] project.
Chat	Displays an interactive chat.
Avatar	Displays an avatar that is used in the proximity-based audio tool.

A.2 Tools

Name	Description
Grid	Changes the layout of elements in the workspace to a grid.
Speaker View	Changes the layout of elements in the workspace to a view where one element—the speaker—is displayed large and other elements are displayed in a filmstrip above the speaker.
Layout Template	Stores the layout of elements in a room and restore the layout later (this only restores the position of existing elements but does not recreate the elements).
Doorway	Allows to listen in on nested rooms by hovering over it with the mouse cursor.
Broadcast	Routes the audio to all rooms.
Whisper	Allows to whisper to other clients in the same room by hovering over their camera feed.
Proximity-based Audio	Changes the audio routing to be based on the location of avatars in a room.
Pedestal	Makes avatars that are close to the pedestal audible to all other clients in a room when using the proximity-based audio.
Recording	Creates recordings of single camera streams.
Room Recording	Creates a recording of the whole workspace of the current room.
Template Store	Enables to store element layouts and restore them later (including recreating the elements).
Highlight	Enables to highlight content WYSIWIS in the workspace by double clicking on it.
Camera Shaker	Shakes the camera stream of clients that are talking loudly in a room.

B VARV CODE EXAMPLE

```
{ "concepts": {
  "element": {
    "schema": {
      "globalHighlighted": "boolean"
    }
  },
  "highlightTool": {
    "schema": {
      "isActive": "boolean"
    },
    "actions": {
      "checkHighlightActive": { "then": [
          { "exists": {
            "concept": "highlightTool",
            "where": {
              "property": "isActive",
              "equals": true
            }
          }},
          { "where": {
            "variable": "exists",
            "equals": true
          }}
      ]},
      "resetGlobalHighlightOnFullscreenClick": {
          "when": { "click": { "concept": "element" }},
          "then": [
            "checkHighlightActive",
            "checkIsFullscreenRoom",
            "resetGlobalHighlight"
          ]
      },
      "resetGlobalHighlight": { "then": [
          { "select": "element" },
          "checkInFullscreenRoom",
          { "set": { "globalHighlighted": false }}
      ]},
      "setGlobalHighlight": {
        "when": { "mousedown": {
          "concept": "contentElement"
        }},
        "then": [
          "checkHighlightActive",
          "checkInFullscreenRoom",
          { "where": {
            "property": "selected",
            "equals": true
          }},
          { "run": "resetGlobalHighlight" },
          { "set": { "globalHighlighted": true }}
        ]
      }
    },
    "extensions": { "inject": [
        "element", "toolElement", "movableMixin",
        "resizableMixin", "droppableMixin"
    ]}
} } }
```

Listing 1: Varv code example of the Highlight tool. The code is slightly simplified for readability.

Varv code is written in the JSON format. It is structured in *concepts*, e.g., a specific tool is its own concept (similar to classes in regular object-oriented programming (OOP)). Each concept consists of a *schema*, *actions*, and *extensions*. The schema defines the properties of a concept. For example, we add the Boolean property `globalHighlighted` to the schema of an `element` to be able to set its highlight value to `true` or `false`. Actions define the interactive behavior of a concept and are trigger-action (when-then) rules. The *when* parts define when an action should be triggered, e.g., when the user clicks on a certain type of concept. This part is optional and actions can also only consist of a then-part. The *then* part defines which action chain should be run, passing an event along from one action to the next. Actions are applied to selections of concept instances (concept instances are similar to objects in OOP). Finally, extensions allow for polymorphism and can be used to inject one concept into another. For example, the `element` concept has properties for its position and size; by injecting the concept into the `highlightTool` it inherits these properties.

Actions in Varv are named and can be used in other action chains. This makes it possible to create a domain-specific language for a certain type of software. In *Mirrorverse*, we created the action `checkInFullscreenRoom` to check whether an element is in the current room that is active in the workspace (the `fullscreenRoom`). Actions such as this one can be used when implementing elements—both in content and tools. Similarly, in the highlight tool, we added the action `checkHighlightActive` which checks whether the highlight tool is active and only continues the action chain if it is.

C ADDING TOOLS TO MIRRORVERSE

Adding new concept types or tools to *Mirrorverse* works similar to reprogramming a tool as described in Section 4.3.5: The user also has to open the Cauldron IDE using the "Edit" button but instead of editing existing code needs to create a package for the new content type or tool and add a Varv concept definition and a Varv template with some boilerplate code (see below for an example). The tool can be used afterwards.

```
{ "concepts": {
  "myNewTool": {
    "schema": {
      // Tool properties here
    },
    "actions": {
      // Tool actions here
    },
    "extensions": { "inject": [
      "element", "toolElement", "movableMixin",
      "resizableMixin", "droppableMixin"
    ]}
} } }
```

Listing 2: Varv boilerplate concept definition code required when adding a new tool.

```
<dom-view-template>
  <varv-template name="myNewToolContent">
    <div class="tool-title">My New Tool</div>
    <!-- Tool content here -->
  </varv-template>
  <varv-template name="myNewToolOptions">
    <!-- Tool options here -->
    <template-ref template-name="toolMirrorOptions">
    </template-ref>
  </varv-template>
</dom-view-template>
```

Listing 3: Varv boilerplate template code required when adding a new tool.