interfacing between such components. Another important category of stream operators facilitate stream *fusion* and *merging*, and they provide the basis for reproducible, correct synchronization across streams. Still other operators enable the *dynamic* construction of computation graphs that can change their structure throughout the execution process, depending on the data flowing through the pipeline. Finally, APIs exist for hierarchically encapsulating a subgraph of multiple components into a single *composite* component. Overall, the runtime, tools, and components provided by *psi* streamline development efforts both at the level of the *component writer*, and at the level of the *application writer*. The same developer can alternate between both roles, utilizing both off-the-shelf components provided in the framework or available from the community in order to assemble their application, while also occasionally writing new components to target specific functions needed by their application. New components can be easily developed and added in a way that shields component authors from the intricacies of the concurrent, coordinated execution environment. We hope the existing set of components will grow into an even larger ecosystem through community contributions, further lowering the barrier to entry for developing multimodal integrative-AI applications.

Implications for HRI

The scope and use-cases of Platform for Situated Intelligence are broad: any application that processes streams of data, and where timing is important, can benefit from its programming models, primitives, and tools. Examples range from analyses of multimodal data all the way to open-world social robots. As such, we believe the framework is particularly well-suited for researchers in HRI. Given the acute needs in this space, a number of related infrastructure and development tools have been developed over the years, such as IrisTK, SSI, MediaPipe, and ROS. IrisTK (**?**) is a Java-based framework that focuses on multiparty face-to-face interaction and social robotics; it provides a set of modules for perception and production, and formalizes the authoring of dialog control around Harel state-charts. The Social Signal Interpretation (SSI) framework (**?**) provides tools that enable synchronized recording, analyzing, and recognizing human behavior in real time. MediaPipe (**?**) enables developers to easily create custom perception pipelines as graphs of modular components, connecting sensors to arbitrary ML inference models and other media processing components.

Most familiar to the HRI community is Robot Operating System (ROS) (**?**), which provides a message-passing infrastructure, an open ecosystem of components, and a set of tools that simplify development of robotics applications. In ROS, each component executes in its own process, and a topology-aware central node handles connections. Since inter-process communication is more costly, this tends to lead to the development of coarser, monolithic components. By contrast, *psi* applications tend to contain a large number of fine-grained components, all residing in the same multi-threaded process, with the *psi* runtime efficiently handling message scheduling and communication. This architecture affords powerful features such as delivery policies, throttling, and back-pressure. Each of these frameworks has beneficial attributes, and all have been successfully used in research. Distinguishing characteristics of *psi* include its programming model which aims to leverage pipeline parallelism and encourages small, light-weight components that are executed in a concurrent yet coordinated fashion, its large set of pre-built primitives for reproducible synchronization and for reasoning about and manipulating temporal streaming data, and its debugging and visualization tools to speed up development. Platform for Situated Intelligence is designed as an extensible framework, and can bridge to and integrate with other ecosystems such as ROS, Python, JavaScript, Unity, etc. One can construct hybrid systems that take advantage of different affordances in different frameworks, e.g., a mobile social robot that uses ROS nodes for navigation and *psi* components for social perception. In addition to what is currently built-in, developers can write their own third-party visualizers and third-party data importers, e.g., for WAV files, mpeg videos, ROS bags, and so on. *psi* applications can execute on a single machine, or can be distributed across multiple machines through remoting capabilities.

We plan to continue to extend the functionality and components available in *psi*, with a particular focus on embodied, physically-situated interaction. We are working on more components and tools for multimodal perception of the situated social context, controllers for generating utterances, intentions, and actions, and realizers for rendering those actions onto the behaviors of a social robot or virtual agent.

Given the challenges around multimodality, integrating multiple technologies, and handling time and data synchronization, it is no surprise that we have still not seen large breakthroughs in the space of complete end-to-end systems that can interact with people in the real world, like we have for the individual component technologies and research that can be conducted offline over large datasets. Enabling people to make breakthroughs on larger integrative systems is what *psi* was built to achieve. Ultimately, we believe lowering the barrier to entry in this space will rest to a large degree on fostering a community of users and contributors, and creating a thriving ecosystem of reusable components. To learn more and get started with the framework, please see https://github.com/microsoft/psi for walkthroughs, samples, and documentation. We invite anyone who is interested to help improve and evolve the platform, and we welcome contributions across the board: from simply using it and filing issues and bugs, to writing and releasing new components, to contributing new features or bug fixes.

Acknowledgements

Platform for Situated Intelligence is the work of a larger team of engineers and researchers. We would like to acknowledge the current members of the *psi* team, which includes Ashley Feniello and Nick Saw, as well as past members: John Elliott, Don Gillett, Anne Loomis Thompson, Mihai Jalobeanu, and Patrick Sweeney. We would also like to thank Eric Horvitz for his contributions and support, as well as our early adopters for their feedback and suggestions.

Voluptas atque aspernatur quos quaerat, deserunt nostrum

natus quibusdam ipsa sit earum harum, libero excepturi doloremque numquam odio ex amet, eligendi nihil