Availability as a Service

Marius Lang, Tom Gross · Human-Computer Interaction Group, University of Bamberg, Germany · *hci@uni-bamberg.de*

Abstract. In Computer-Supported Cooperative Work, respecting the mutual availability in teams to reduce disturbing interruptions while keeping team members in touch is very important. Detecting team members' availability is possible, but complex. In this paper we present the Availability-as-a-Service microservice that provides a generic interface to eye trackers, encapsulates complex calculations of availability based on eye tracking data, and provides the team members' availability status to other software components.

Keywords. availability, disruption, eye tracking, software service architecture.

1 Introduction

Humans continuously interact with other humans or computer systems. These interactions lead to interruptions. Whether such interruptions are perceived as supportive or disruptive depends on humans' availability, defined as whether the user is currently engaged in a task or available for collaboration, and the timing of the interruption (Adamczyk & Bailey 2004; Hirsch *et al.* 2024; Ohly & Bastin 2023; Rick *et al.* 2024).

Existing approaches to availability estimations rely on contextual factors such as calendar data, device usage, or environmental cues (Horvitz *et al.* 2003). In particular, they fail to account for cognitive load, which strongly influences interruption costs and task performance (Bailey & Iqbal 2008; Hirsch *et al.* 2024).

Cognitive load refers to "the relative demand imposed by a particular task, in terms of mental resources required. Also called mental load; mental workload." (American Psychological Association 2018). Eye tracking, particularly pupillometry, provides insights into a human's cognitive load (Van Der Wel & Van Steenbergen 2018). By leveraging this signal, availability estimation systems can reflect the human's cognitive condition and adapt interaction timing accordingly. While applying pupillometry to detect cognitive load in order to find opportune moments for interruptions is promising, it is also complex and tedious to implement from scratch.

We present Availability as a Service (AaaS), a modular platform that leverages pupillometry to estimate human availability and to provide third-party systems with a standardised interface for querying it. A central element is the integration of eye tracking signals into the process of availability estimation. By analysing pupillometry data, the platform can assess cognitive load in real time and use it to determine human availability. This enables the system to provide availability estimations for interruptions explicitly based on real-time measures of cognitive

Marius Lang, Tom Gross. 2026. Availability as a Service. In: *Proceedings of the 24th EUSSET Conference on Computer-Supported Cooperative Work (ECSCW) – Posters and Demos*, Reports of the European Society for Socially Embedded Technologies. ISSN: 2510-2591 https://doi.org/10.48340/ecscw2026_121

load. The platform is built on a service-oriented architecture following Software as a Service (SaaS) and microservice principles and supports both centralised and decentralised deployment models.

2 Related Work

Designing a platform that estimates human availability requires grounding in the behavioural science of interruptions and the engineering principles that govern service-oriented systems. We review existing work across three areas: the cognitive effects of interruptions, prior approaches to availability estimation, and the architectural paradigms that inform the design of AaaS.

In Human-Computer Interaction (HCI), interruptions are defined as events where a user's ongoing primary task is temporarily paused to address a secondary task (Gross & Von Kalben 2023; Trafton *et al.* 2003). Interruptions are common in digital environments. Users encounter emails, instant messages, phone calls, system alerts, and other sources of distraction (Rick *et al.* 2024). Such interruptions often cause a disruption. Consequently, managing interruptions has become a key concern in the design of systems.

In early research, interruption management relied heavily on cues external to the user, such as the time of day, location, or calendar events, to infer a user's availability (Fetter *et al.* 2018; Gross & Prinz 2003; Gross & Prinz 2004; Horvitz *et al.* 2003). Recent work in HCI has shifted focus from static context models to dynamic mental models, emphasising the user's cognitive load at the centre of availability estimation systems. Cognitive load is suitable for breakpoint detection because breakpoints are characterised by changes in the level of cognitive load experienced during a task (Bailey & Iqbal 2008). Measuring it continuously and unobtrusively is a crucial step in estimating availability for interruptions. Pupillometry, the systematic measurement of pupil diameter changes over time, provides this capability (Mathôt & Vilotijević 2022). As cognitive load increases, attention regulation in the autonomic nervous system produces a measurable increase in pupil diameter (Van Der Wel & Van Steenbergen 2018). This physiological response has been shown to reflect cognitive load across a range of tasks without requiring explicit input from the user (Zagermann *et al.* 2018). Therefore, it is well-suited as the core signal for an availability estimation platform.

With pupillometry established as a suitable signal for cognitive load measurement, it is worth examining how existing systems have approached availability estimation in practice. TeamMeetingArranger by Maleck and Gross (2025) advances this by incorporating pupillometry to assess cognitive load, scheduling meetings only when all participants fall below a dynamic threshold. Yet it remains a standalone solution scoped to a meeting arrangement. It does not provide an interoperable interface through which third-party systems could query or react to availability. AaaS is designed to address this gap.

Addressing this gap requires a platform that supports standardised access, multi-tenant operation, and low integration effort for third-party systems, which points towards a service-oriented delivery model. Software as a Service (SaaS) is defined as "a business and software delivery model that enables organisations to offer their solutions in a low-friction, service-centric model that maximises value for customers and providers" (Golding 2024). Its properties of scalability, shared resource utilisation, and standardised access through network interfaces address these requirements.

Microservices represent an architectural paradigm of SaaS in which systems are composed of small, autonomous services that implement distinct business capabilities and communicate through well-defined interfaces (Newman 2015). Unlike monolithic architectures, this approach decentralises control and execution, enabling individual services to evolve, scale, and fail independently (Bucchiarone *et al.* 2020).

Our AaaS leverages on the strengths of the SaaS and microservice principles, which provide our architectural foundation. It exposes availability estimation as a self-contained service accessible by multiple third-party systems through a standardised interface. By enabling independent development, deployment, and scaling, AaaS supports modularity, extensibility, and fault isolation, which are essential for implementing AaaS as a flexible and evolvable platform.

3 AaaS Concept

AaaS is based on a proper microservice concept: a dedicated service exposes and manages the availability status of its users. Similar to how SaaS provides access to software functionality over a network, AaaS provides structured access to users' availability. This abstraction allows third-party systems to query and respond to availability without any knowledge of the underlying estimation method, sensor hardware, or user context.

To implement this idea effectively, AaaS adheres to the architectural principles of SaaS. It is designed as a self-contained software unit that encapsulates all the necessary components into a single deployable application. This ensures minimal setup effort and eliminates the need for additional services. Only one software package is required to deliver the full range of functionalities that AaaS offers.

Since the AaaS concept builds on SaaS, it is necessary to determine what a tenant is in this context. In AaaS, a tenant refers to a logically separated unit of ownership within the system. This represents an individual user whose availability is managed by AaaS. Third-party systems can act on behalf of that tenant to query availability information.

AaaS estimates availability through a structured processing pipeline (see Figure 1), beginning with the collection of raw pupil diameter data from the eye tracking device. The raw signal passes through a pre-processing step to filter noise. At the core of AaaS lies the logic for estimating a user's availability. This layer

determines how raw eye tracking data is interpreted to infer whether a user is currently available for an interruption. The design of this layer is flexible and extendable, supporting different algorithms based on the use case.

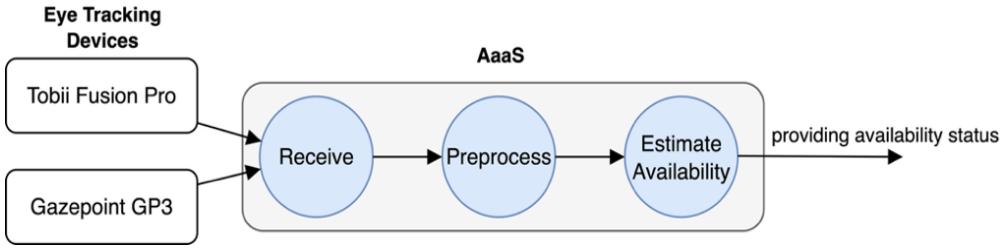

Figure 1. Data flow diagram of the availability estimation pipeline. Eye tracking devices (Tobii Pro Spectrum and Gazepoint GP3) provide real-time measurement of pupil diameters, which are received, pre-processed, and analysed within AaaS to determine the user's availability.

Instead of relying on a single, hard-coded method for availability estimation, AaaS supports multiple estimation strategies that can be selected for each user at runtime. These are implemented as microservices, each encapsulating its logic for interpreting eye tracking data. This approach enables estimating availability using cognitive-load-based thresholds, time-based rules, user feedback, or hybrid models that combine multiple indicators. To encourage user-specific adaptation, AaaS enables external developers to register their availability estimation strategies. Developers can integrate their own strategies into the system and selected at runtime, ensuring consistent communication regardless of the underlying estimation logic. Through this modular architecture, AaaS decouples the availability estimation logic from the rest of the service, enabling users to customise it to their own needs.

For the developers' convenience, AaaS provides two implementations of availability estimation strategies: one that segments work into cognitive-load-adaptive phases with explicit user feedback to refine phase duration, and one that evaluates interruption requests on demand by comparing the user's current pupil-diameter pattern against historically well-rated interruption moments.

AaaS offers user availability to third-party systems through three interaction models: on-demand queries that return the current availability status; a subscription-based push mechanism that notifies third-party systems immediately upon availability changes; and an interruption registration model, in which third-party systems declare their intent to interrupt, and either receive immediate permission or wait in a queue until availability permits. Together, these models ensure that third-party systems can both observe and act on availability in a consistent and coordinated manner.

4 AaaS Implementation

AaaS encapsulates the internal responsibilities into multiple microservices as a basis for its modularity, flexibility, and the separation of concerns, and provides a

generic interface to eye tracking devices as well as an API gateway for providing its services to third-party systems (see Figure 2).

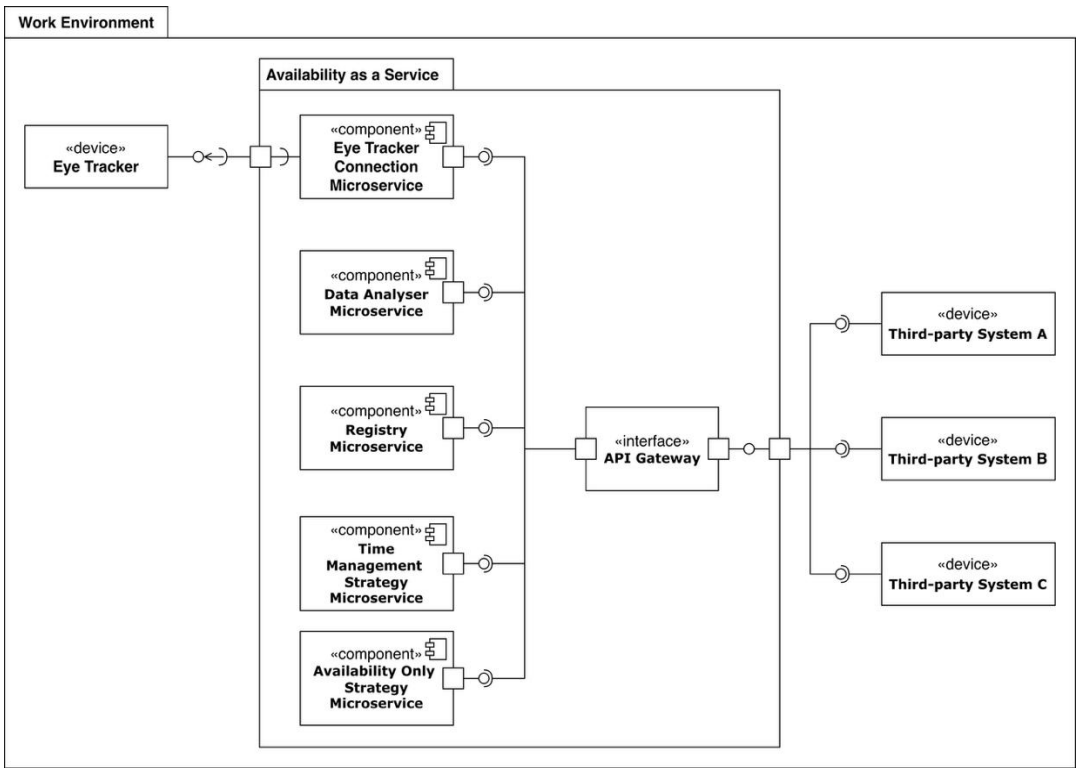

Figure 2. UML component diagram illustrating the architecture of AaaS. All external access is routed through the API Gateway. Internal microservices handle eye tracking data reception, analysis, and availability estimation.

An API Gateway serves as the primary entry point for all external requests. It provides a unified interface while shielding third-party systems from the underlying complexity. Behind the API Gateway, a set of specialised microservices performs the core functions: the `Eye Tracker Connection` Microservice receives raw eye tracking data, the `Data Analyser` Microservice preprocesses and analyses it, and one of the default availability estimation strategy microservices, either the `Time Management Strategy` Microservice or the `Availability Only Strategy` Microservice, estimates the user's availability. The `Registry` Microservice maintains the registration state of the availability estimation strategy microservices. All microservices are implemented in Python (version 3.10) using the FastAPI framework (version 0.115.12).

In our implementation, AaaS connects to a Tobii Pro Spectrum with 1200 Hz via the `tobii-research` Python library (version 1.11.0), and to the Gazepoint GP3 via the `Gazepoint Open Standard API` over TCP.

The API Gateway is implemented in Python and exposes both RESTful HTTP endpoints and WebSocket endpoints, enabling third-party systems to query availability on demand and subscribe to updates.

5 Evaluation

During the OpenLab-Night of the HCI-Chair at the University of Bamberg, AaaS was made available to nineteen individuals of varied age, gender, and background. Although this was not a formal user study, it confirmed that the platform operated reliably under repeated short-term usage. A subsequent log file analysis identified minor bugs, which have been resolved. A formal evaluation of the estimation strategies' accuracy and long-term performance remains a subject for future work. Despite the fact that the system captures and analyses the users' fixations and saccades in detail, privacy was not a concern for the participants. This might be due to software architecture of AaaS that encapsulates all eye tracking data and allows to only share abstract and processed information to other software components.

6 Conclusions

This work presented Availability as a Service (AaaS), a platform that leverages pupillometry to estimate cognitive load and expose availability status to third-party systems through a unified interface. AaaS meets its core architectural requirements, demonstrating reliable operation and confirming that the modular microservice architecture enables new availability estimation to be integrated with minimal effort. Future work should address remaining gaps in authentication and access control, and evaluate estimation strategies through user experiments measuring estimation accuracy, perceived interruption and disruption, and task resumption time. Furthermore, additional physiological data sources such as electrodermal activity or heart rate variability should be explored.

Acknowledgements

We thank the members of the Cooperative Media Lab of the University Bamberg.

References

Adamczyk, Piotr D. and Bailey, Brian P. 2004. If not now, when?: the effects of interruption at different moments within task execution. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2004* (Apr. 24-29, Vienna, Austria). ACM. pp. 271-278.

American Psychological Association. *Cognitive Load*. American Psychological Association, https://dictionary.apa.org/cognitive-load, 2018. (Accessed 31/08/25).

Bailey, Brian P. and Iqbal, Shamsi T. Jan. 2008. Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. *ACM Transactions on Computer-Human Interaction* 14. pp. 1-28.

Bucchiarone, Antonio, Dragoni, Nicola, Dustdar, Schahram, Lago, Patricia, Mazzara, Manuel, Rivera, Victor and Sadovykh, Andrey, eds. 2020. *Microservices: Science and Engineering*. Springer, Cham.

Fetter, Mirko, Mueller, Anna-Lena, Vasilyev, Petr, Barth, Laura Marie and Gross, Tom. 2018. Towards a Better Understanding of Availability and Interruptibility with Mobile Availability Probes. In *Proceedings of the 16th European Conference on Computer-Supported Cooperative Work - Exploratory Papers - ECSCW 2018* (June 4-8, Nancy, France). Reports of the European Society for Socially Embedded Technologies, EUSSET, Siegen, Germany. pp. 14:1-18.

Golding, Tod. 2024. *Building Multi-Tenant SaaS Architectures*. O'Reilly Media, Inc, Sebastopol, California, USA.

Gross, Tom and Prinz, Wolfgang. 2003. Awareness in Context: A Light-Weight Approach. In *Proceedings of the Eights European Conference on Computer-Supported Cooperative Work - ECSCW 2003* (Sept. 14-18, Helsinki, Finland). Kluwer Academic Publishers, Dordrecht. pp. 295-314.

Gross, Tom and Prinz, Wolfgang. Aug. 2004. Modelling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. *Computer Supported Cooperative Work: The Journal of Collaborative Computing* 13, 3-4. pp. 283-303.

Gross, Tom and Von Kalben, Michael (2023). *A Literature Review on Positive and Negative Effects of Interruptions and Implications for Design*. In Nocera, J.A., Lárusdóttir, M.K., Petrie, H., Piccinno, A. and Winckler, M., eds. Human-Computer Interaction – INTERACT 2023. Springer, Cham. pp. 373-379.

Hirsch, Patricia, Moretti, Luca, Leichtmann, Benedikt, Koch, Iring and Nitsch, Verena. Jan. 2024. Opportune moments for task interruptions: examining the cognitive mechanisms underlying interruption-timing effects. *Frontiers in Psychology* 15. pp. 1465323.

Horvitz, Eric, Kadie, Carl, Paek, Tim and Hovel, David. Mar. 2003. Models of attention in computing and communication: from principles to applications. *Communications of the ACM* 46. pp. 52-59.

Maleck, Moritz and Gross, Tom. 2025. TeamMeetingArranger: A Less Disruptive Way of “Do You Have a Minute?”. In *Proceedings of Intelligent Human Computer Interaction 2024 – IHCI 2024* (Nov. 13, Twente, The Netherlands). Springer. pp. 54–65.

Mathôt, Sebastiaan and Vilotijević, Ana. Aug. 2022. Methods in cognitive pupillometry: Design, preprocessing, and statistical analysis. *Behavior Research Methods* 55, 6.

Newman, Sam. 2015. *Building microservices: designing fine-grained systems*. O'Reilly Media, Beijing Sebastopol, CA.

Ohly, Sandra and Bastin, Luca. Jan. 2023. Effects of task interruptions caused by notifications from communication applications on strain and performance. *Journal of Occupational Health* 65. pp. e12408.

Rick, Vera B., Brandl, Christopher, Knispel, Jens, Slavchova, Veneta, Arling, Viktoria, Mertens, Alexander and Nitsch, Verena. Apr. 2024. What really bothers us about work interruptions? Investigating the characteristics of work interruptions and their effects on office workers. *Work & Stress* 38, 2. pp. 157-181.

Trafton, J.Gregory, Altmann, Erik M, Brock, Derek P and Mintz, Farilee E. May 2003. Preparing to resume an interrupted task: effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies* 58. pp. 583-603.

Van Der Wel, Pauline and Van Steenbergen, Henk. Feb. 2018. Pupil dilation as an index of effort in cognitive control tasks: A review. *Psychon Bull Rev* 25, 6. pp. 2005-2015.

Zagermann, Johannes, Pfeil, Ulrike and Reiterer, Harald. 2018. Studying Eye Movements as a Basis for Measuring Cognitive Load. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2018* (Apr. 21-26, Montreal, QC, Canada). ACM. pp. 1-6.