bRIGHT – A Framework for Capturing and Adapting to Context for User-Centered Design

Rukman Senanayake and Grit Denker(✉)

SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA
{rukman.senanayake,grit.denker}@sri.com

Abstract. The ability to create and maintain a highly accurate model of an end user's context is an extremely useful feature. Achieving this ability poses many challenges, especially since a great degree of partial information and uncertainty is involved in capturing the user's context. The bRIGHT human-computer interaction (HCI) framework and workstation address these challenges by creating a highly accurate context model of a user engaging with a computer system. In this paper we discuss the architectural design of bRIGHT, which addresses performance and scalability to build accurate user context models, and the benefits we expect from this improved version. We also discuss technological advances in other related fields that influenced our decision-making.

Keywords: User context modeling · Run-time adaptation and automation · User assistance · Context-aware automation · Context-aware filtering · Context-aware prediction · Cognitive autofill · User-centered design architecture · User-centered system

1 Introduction

The term context-awareness was coined with the rise of ubiquitous computing [1] and quickly became an important topic in human computer interaction (HCI) [2]. Context of use is fundamentally important in providing meaningful, efficient, effective, and adaptive user experiences (UX). Understanding the user and her context is the key enabling factor in tailoring to current needs and making user experiences relevant, and it also opens the path to dynamically adaptive system designs.

The word context covers a broad set of meanings in the UX and conventional HCI arenas. For the purpose of this paper, and with respect to the bRIGHT system (Fig. 1) and its current capabilities, we are focused on the aspects of the user's context that can be observed and recorded in terms of interactions with a computer system. There are many other aspects to a user's context [3–5]: location; physical vs. logical, societal relationships and impacts (e.g., a pandemic changing how a user works); environmental impacts; and interface interactions with a computer or a machine. We are focused on modeling the subset of the user's context that can be directly observed in terms of their engagement with the system. The research questions we are investigating are:

Published by Springer Nature Switzerland AG 2022
C. Ardito et al. (Eds.): INTERACT 2021, LNCS 13198, pp. 158–173, 2022.
https://doi.org/10.1007/978-3-030-98388-8_15

- Can an accurate and detailed user context model positively and ongoingly enhance UX during system operation?
- What degree of accuracy is required in a context model for achieving transformational gains in UX?

In particular, bRIGHT does **not** aim to improve by adapting the user interface (UI) itself; rather, it works with given UIs and improves the UX of the users in those UIs (cf. Sect. 2). Currently, bRIGHT does not include technology to detect and integrate events in external, environmental objects, but the framework allows for such extensions that we plan for the future.

bRIGHT is a sensing, modeling, and analysis framework that allows accurate modeling and adaptation to a user's context. The *bRIGHT context models* store each user action in a machine-processable and meaningful way that can be queried and used in algorithms to determine user interest and provide predictions and assistance to the user. bRIGHT allows rich and meaningful context and contains information at an abstraction level to permit semantically equivalent statements such as, "After the company's West coast network security administrator read an email from her East coast colleague about an ongoing attack on the company's network, she proactively redirected identified traffic through the company's honeynets." Observational approaches use less-rich context data such as keystrokes, heat maps, or gaze patterns.

bRIGHT is beneficial for human-centered software engineering (HCSE) in two ways. In the short term, user-centered design processes can make use of bRIGHT's user context (and future cognitive) models to design systems that can adapt to new requirements or changes in use. It is common for user modeling and context to be used in the design and implementation of interfaces and interface components. But approaches in which the user's context is tracked and modeled in real time and then used to adapt the entire user experience are still rare. bRIGHT was designed and developed to model a user's context, maintain the context model as accurately as possible, and evolve the UX by offering proactive assistance to the user. The scope of our work covers the development of a research framework that captures the user's context and allows formal modeling of mechanisms and techniques to update the model accurately. We also include the ability to reason about the dynamics in the context model, and to investigate software and hardware designs that would enable adaptive UX and properly harness the synergy of these features. In addition, bRIGHT's user context models do not forget; they represent a time-machine-like record of all observed user actions. As such, these models are useful in the short- and long term because they provide a basis for context of use and evolutionary trends.

Section 2 provides some related work and Sect. 3 gives an overview of the bRIGHT system. A use case is presented in Sect. 4. Details of the approach and status of development are provided in Sect. 5. Section 6 summarizes the next steps for bRIGHT development and application and contains our conclusions.

2 Related Work

With increasing regularity, humans are interacting with autonomous systems and also with systems powered by artificial intelligence (AI). In such interactions, the systems'

ability to observe, model, and leverage the user's context becomes vital in improving system effectiveness and efficacy. The context is pivotal to the decision-making process of the user; decision-making does not happen in a vacuum and is almost always grounded in user context. If autonomous systems or AI-based systems are capable of tracking and reasoning about the user's context, then they can also play a role in providing the right kind of assistance at the right time and place.

To characterize bRIGHT with respect to other context-aware systems, and following the nomenclature in [4], bRIGHT is a context-server based approach, and in particular a context-aware framework, because it allows extension to specific context. bRIGHT uses a semantic type system to model context that is similar to ontological approaches, but does not use the full power of ontological relationships. bRIGHT's type system records for each contextual model instance the context type and value, time stamp, and the (sensor) source and confidence of the contextual instance. Context processing is done in bRIGHT in various ways: We use rules for interest modeling as well as context interpretation (e.g., knowledge about domain-specific classes like Internet Protocol (IP) address, types of attacks, and so on in the security domain). bRIGHT is not currently handling security and privacy yet, but in a separate project we have implemented an ontology-based policy reasoner for privacy and security that could easily be integrated with bRIGHT. Historical context data is by default integrated into bRIGHT, since bRIGHT's context model does not forget and keeps the entire user context history.

It is common for user modeling and their context to be used in the design and implementation of interfaces and interface components. For example, (contextual) personas [6] are an approach to enrich the communication between designer and developer, and contextual personas [7] include aspects of the digital work environment. Value-Based Requirements Engineering (VBRE) [8] makes users' values explicit through analysis with a reference taxonomy. The taxonomy contains concepts such as values, motivations, and emotional reactions, all of which are important in decision making. VBRE can be used in requirements engineering to support user-centric design (UCD) (e.g., as done in [9] for a health decision support system). These approaches are used in the requirements or design phase of systems. bRIGHT's approach is different in that it is used during system operation. bRIGHT observes the user, builds a contextual model of the user's interest, and uses that model to dynamically adopt the UX at runtime. This use of bRIGHT is in line with work that aims to improve processes through integration of machine learning (ML) [10] and benefits user experiences in Web Internet of Things (IoT) applications [11]. bRIGHT's implementation is intentionally generic so that it can serve as a basis for many possible applications.

Currently, bRIGHT's focus is on modeling user interactions with a set of computer applications to create an accurate user context model during operation. bRIGHT uses its context model for a variety of adaptation techniques such as "right information at the right time" or "workflow/task automation" or "contextual auto-fill from context model." Section 4 provides examples of those adaptation techniques.

Other approaches aim to improve usability through adaptive behavior of the UI. For example, [12] provides users with a minimal feature-set and an optimal layout based on the context of use. They are using Role-Based UI Simplification (RBUIS) based on

a-priori statically defined user roles and tasks. Role-based UI models support feature-set minimization by assigning roles to task models and layout optimization through workflows that represent adaptive UI behavior visually and through code. The main difference to our approach is that the roles and corresponding adaptations are determined a-priori, whereas our approach models the user at runtime and determines adaptations very specifically to what the user is and has been looking at and what actions the user has done so far. The adaptations are thus very tuned to the current user's context. As such, bRIGHT's approach could be used in conjunction with RBUIS or similar UI adaptation frameworks.

Another line of research focuses on enabling end users to easily and autonomously personalize the behavior of their applications. For example, [5] presents an approach that allows end users without programming experience to customize the context-dependent behavior of their IoT applications through the specification of trigger-action rules. The goal is to support the dynamic creation and execution of personalized application versions that are more suitable for users' needs in specific contexts of use. bRIGHT's context model is accessible through programming interfaces and could be used for end user development activities as described in [4], but we have not investigated this avenue of research.

bRIGHT also does not attempt to automate UI generation adapted to a person's devices, tasks, preferences, or abilities, like SUPPLE [13], which formally defines interface generation as an optimization problem that is feasible for a particular class of cost functions. The notions of cost functions for adaptations as used in SUPPLE would be interesting to investigate in the future for bRIGHT as a quantitative measure of value added by automations provided by bRIGHT. For example, one of the cost functions in [13] models a person's ability to control the pointer and allows SUPPLE to generate user interfaces adapted to unusual interaction techniques or abilities, such as an input jittery eye tracker or a user's limited range of motion due to a motor impairment. bRIGHT's automation techniques could be beneficial in some of these circumstances (e.g., avoiding the need to control the pointer by providing pre-filled choices that users can confirm through other means (voice, return key).

In the future, it would also be interesting to integrate the foundational context language ContextML presented in [14] with bRIGHT's context model. This would enable the use of the bRIGHT's context model as part of the Model-Driven UI Development framework of [14] (assuming appropriate framework APIs). bRIGHT's context model is rich with domain-specific context and could be beneficial to more user- and context-specific adaptations.

The context model built by bRIGHT is similar in many ways to context models defined using ontological approaches such as [15–17]. Using ontologies allows us to use general concepts to build a basic context model that has the necessary extensibility to support domain-specific concepts. The user-profiling ontology developed by Skillen [15] and COBRA-ONT developed by Chen et al. [16] are extensive, and we will consider them for integration into future bRIGHT developments.

3 Overview of bRIGHT

We are using bRIGHT (Fig. 1) to focus on understanding what type of hardware and software design can lead to systems that accurately model user context and use the model and its dynamics to capture user's needs and interests. This information can be used to help the user in a more efficient and effective way. In addition, our approach supports the study of context of use, long-term evolutionary trends, and their impact on the user.

Fig. 1. A bRIGHT workstation consisting of a touch table with proximity detection for positioning controls under the hands of users. The center monitor has a gaze tracking system mounted on the bottom that records what the user is looking at. The user is working with a cybersecurity application that is rendered on the center monitor. The left monitor visualizes the recorded raw data regarding the user's interactions with the application (what the user did) and the user's gaze (what the user looked at). The right monitor visualizes the generated user context model that represents the user's interest at any given time.

We have used bRIGHT as an HCI research framework and as the backbone technology for future human-centered, resilient, and adaptive system designs. In a military application, we used bRIGHT as an HCI framework to study opportunities to integrate AI into counter unmanned aerial systems in simulated environments. We instrumented the simulation software to record user activities and conducted human subject research (HSR) using complementary measures such as video recording of user behaviors and screens, physiological measurements of pulse waveforms, annotations of recordings, task performance measures, interviews, and self-assessment. Analysis of the data collected by bRIGHT helped identify possible optimizations for the counter unmanned aerial simulation system via contextual filtering and cognitive autofill; these made user interface interactions easier by auto-filling relevant data from short-term memory.

We also used bRIGHT as a backbone technology framework in cybersecurity applications and demonstrated user context models dynamically adapting and enabling context-based filtering and task automation [18, 19] as shown in the next section.

4 bRIGHT Runtime Adaptation – A Use Case

We are illustrating bRIGHT's automation adaptations at runtime with a use case from the cybersecurity background. SRI has developed several cybersecurity analysis tools that were instrumented with bRIGHT: BotHunter[1] is a malware detection and analysis tool and Infected America is a tool to visualize IP reputation data. Typically, a network operator or security specialist would use these and other tools to help in the analysis of network status and investigate potential security threats. Since these tools are bRIGHT-enabled, user interactions with these tools are observed and recorded in a user context model. We will show in a step-by-step use case how a context model is used to provide automation on the fly and improve the UX.

Figure 2, 3, 4, 5, and 6 show a sequence of actions performed by the user in the BotHunter malware detection and analysis tool. Figure 2 shows the main Bothunter interface that provides several analytical dashboard options. Users have the option to see forensic graphs that summarize infection status per IP address. Other dashboard panels summarize infection patterns or show the infection profiles in a table format with detailed meta-information per infection. Finally, a table of external interactions for each IP is available. For the use case, the network operator will use the "Forensic Confidence Composition Graph" to understand how different IP addresses in his network have been attacked and the "External Interactions" table to understand with what external IPs his infected devices have communicated.

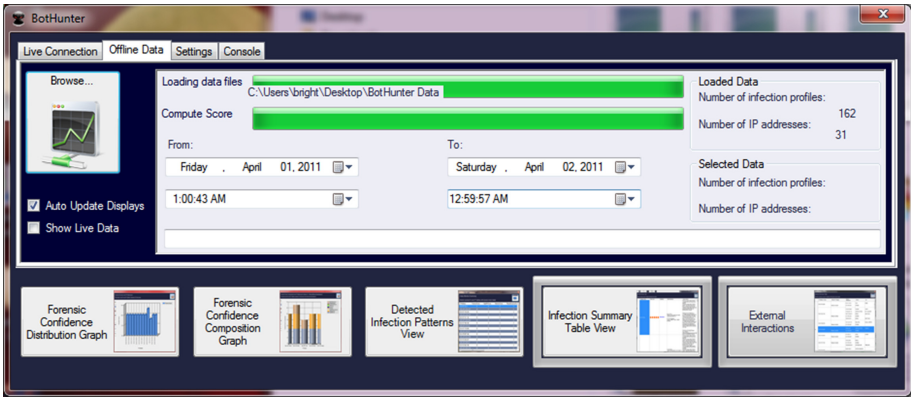

Fig. 2. BotHunter main UI with several analytical dashboard options. The cybersecurity application BotHunter is bRIGHT-instrumented. As the user interacts with BotHunter (and other applications), bRIGHT creates the user context model.

[1] https://en.wikipedia.org/wiki/BotHunter and http://www.bothunter.net/about.html.

Opening up the "Forensic Confidence Composition Graph" results in Fig. 3. For each IP in the operator's network, it shows a color-coded summary of the different types of evidence classes over IP addresses, such as scores for RBN (Russian Business Network), RepeatScanner and DNSCheckIn among others.

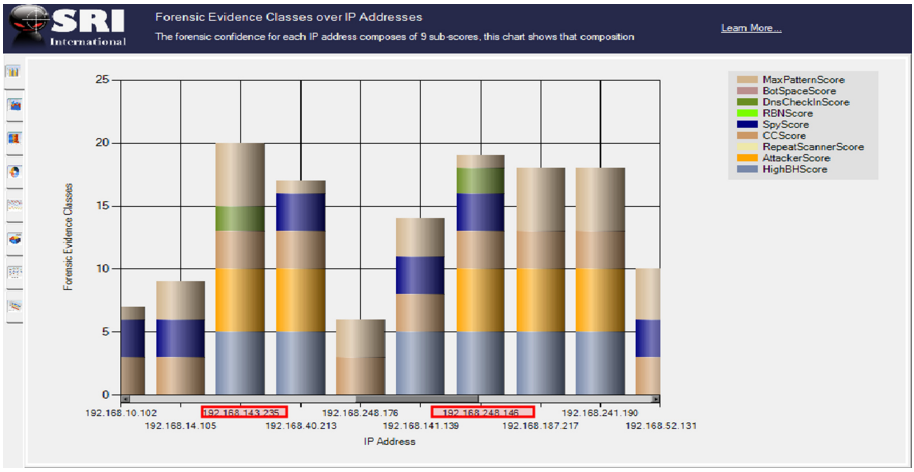

Fig. 3. The Forensic Evidence Classes graph shows for each IP and summary of the kinds of evidence that was collected. The graph is zoomable and clickable so that more details of evidence can be explored by the user.

The network operator is interested in two of the IPs, namely 192.168.143.235 and 192.168.248.146 each of which represents internal IP addresses (PCs or servers) and would like to better understand what is going on with these assets. Double clicking the bar graphs of these two IP addresses opens up new dialog windows that show so-called infection profiles for each of them (Fig. 4 shows an infection profile for 192.168.143.235).

The user is also looking at infection profiles of the other IP addresses of interest. Then the user looks at daily infection summaries per IP in BotHunter to get an idea of how strong the evidence is for detected patterns. For example, the user looks at the "Botnet Infection" information and forensic confidence for IP 192.168.143.235 as shown in Fig. 5.

Finally, the user clicks on the "External Interactions" button to understand what external IPs his network nodes communicated with. This is shown in the External Interactions Table (Fig. 6). The information about communications with external IPs is useful for another cybersecurity application in which the user can get more information about those external IPs and whether they are known to be malicious. All this helps the user to get a complete picture of the situation and decide what actions are necessary to counter a potential attack.

The user's interactions with BotHunter have been recorded in a contextual model that is illustrated in Fig. 7.

Fig. 4. Infection Profile View for one of the chosen IP addresses that are of interest to the user. Each infection profile view may have several infection profiles, each consisting of a list of icons that symbolize the various evidence classes (e.g., inbound attack, egg download, connection to malicious Command and Control (C&C) Server, connection to Russian Business Network (RBN), or outgoing attack)). Infection Profiles provides a quick overview for the operator what has been going on with the network asset in question and where the attack stands. This will enable the user to quickly decide where to put his attention and act on isolating network nodes to limit the damage and contain the exposure.

The user next launches the Infected America application (shown in Fig. 8) to determine the IP reputation data of one of the external IP addresses to which the system associated with IP 192.168.143.235 is connected. At this point, a set of external IP addresses in the user's context model can be used for pre-populating this field. bRIGHT's context model then sends this list of IP addresses to the Infected America application to provide the user contextual auto-fill entries that match both the type (IP address) and the fact that they need to be external addresses and not internal ones (e.g., 192.168.248.146 and 192.168.143.235), since Infected America is used for obtaining information about external IP addresses and their historical behavior, not internal ones.

Infection summary for each IP address with dynamic level of detail

	IP Address	Forensic Confidence	Detected Patterns	Pattern Description
	192.168.1.102	11	Malware Coordination	This system has been observed conducting communications that appear related to malware command and control.
	192.168.10.102	7	Malware Coordination	This system has been observed conducting communications that appear related to malware command and control.
	192.168.14.105	9	Malware Download	This system has downloaded potentially malicious software from an external untrusted location.
	192.168.141.139	14	Spyware	This system is interacting with external hosts using known spyware dialog patterns.
▶	192.168.143.235	20	Botnet Infection	This system has been successfully infected via a remote attack. It has uploaded malicious software, has coordinated with its control server, and is now spreading malware across the Internet.
	192.168.15.15	4	Malware Coordination	This system has been observed conducting communications that appear related to malware command and control.
	192.168.162.142	10	Sasser	This system has been successfully infected with a Sasser-like malware application.

Fig. 5. Detected Infection Patterns View shows for each IP the kind of detected patterns such as malware coordination, spyware and malicious download and how much forensic evidence was collected for each pattern along with other meta data

External Interactions Table

Map

Infection summary for each IP address with dynamic level of detail

IP Address of Victim	External Interactions	External IP Category	ISP	Domain	Country	City
	98.172.183.8	-Uncategorized-	COX COMMUNICATIONS	COX.NET	UNITED STATES	OKLAHOMA CITY
	60.190.223.75	Malware Propagator	SHAOXING TELECOM BUREAU	YZTRADECN.COM	CHINA	BEIJING
	122.224.6.164	Malicous Site, Mail Abuser, Malware Controller	SHAOXING TELECOM BUREAU	YZTRADECN.COM	CHINA	BEIJING
192.168.143.235	195.88.191.59	Malware Propagator	BIGNESS GROUP LTD. NETWORK	PTRZONEZ.COM	RUSSIAN FEDERATION	-
	70.166.97.88	-Uncategorized-	COX COMMUNICATIONS	COX.NET	UNITED STATES	PHOENIX
	60.190.223.75	Malware Propagator	SHAOXING TELECOM BUREAU	YZTRADECN.COM	CHINA	BEIJING
	122.224.6.164	Malicous Site, Mail Abuser, Malware Controller	SHAOXING TELECOM BUREAU	YZTRADECN.COM	CHINA	BEIJING
192.168.248.146	61.147.99.179	Malware Propagator	CHINANET JIANGSU PROVINCE NETWORK	163DATA.COM.CN	CHINA	BEIJING
	174.123.157.154	Malicous Site, Mail Abuser, Malware Controller	THEPLANET.COM INTERNET SERVICES INC	THEPLANET.COM	UNITED STATES	DALLAS
	64.38.232.180	Malicous Site, Mail Abuser, Malware Controller	DOMAIN DEVELOPMENT	-	UNITED STATES	CALABASAS
	83.133.119.197	Malicous Site, Mail Abuser, Malware Controller	LNCDE-GREATNET-NEWMEDIA	GREATNET.DE	GERMANY	-

Fig. 6. External Interactions Table shows for each local IP to which external IPs is has connected. This information is useful as it allows the user to now broaden the analysis outside of his own network and further investigate attacking nodes from the wider internet and decide mitigation strategies (such as taking the local IP off the network and rerouting traffic to that node to a honey net to further understand attackers).

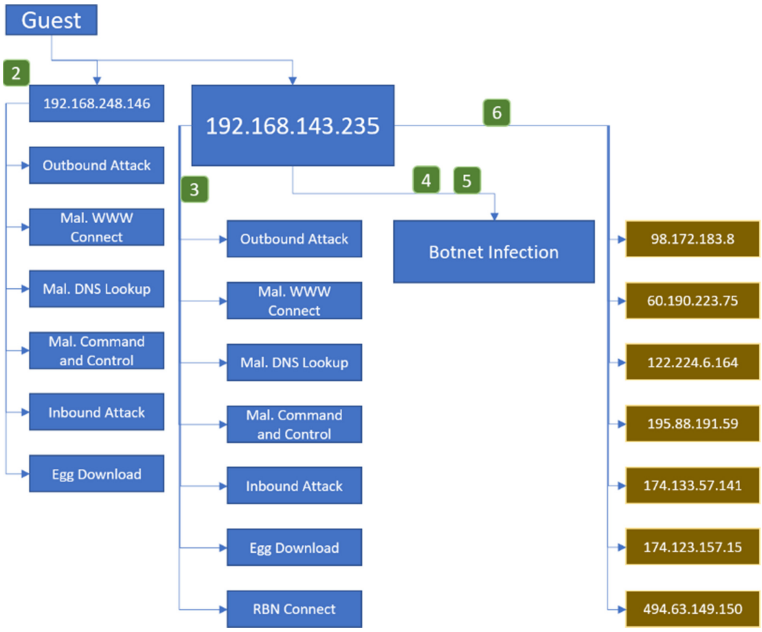

Fig. 7. Context model of a user interacting with BotHunter.

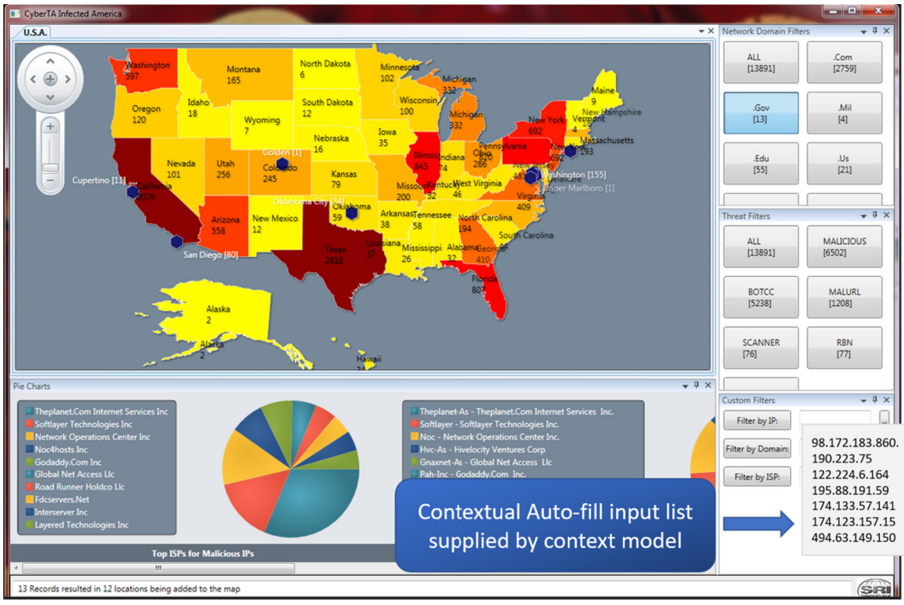

Fig. 8. Infected America, another bRIGHT-instrumented cybersecurity application.

bRIGHT's context model is a result of only a few minutes of user interaction with bRIGHT. This means that bRIGHT can accurately determine interest even with limited user data. In this way, bRIGHT's modeling approach differs from data-hungry modeling approaches such as machine learning to dynamically determine user interest with high accuracy.

5 Approach

In this section, we describe the technology of bRIGHT's design and illustrate how it impacts our research methodology from a user-centered design perspective.

The bRIGHT system is designed and developed to meet certain requirements. Chief among these is the ability to handle high volumes of raw signal information from gaze tracking, multi-touch, and biometric (face recognition, iris scanning) sensors and the ability to scale to support large groups of users working together to solve complex problems. bRIGHT's framework has support for the Kafka streaming server to introduce an integration end point into bRIGHT that is standardized and widely popular.

The use of open-source APIs was motivated by performance considerations and ease of deployment in various labs.

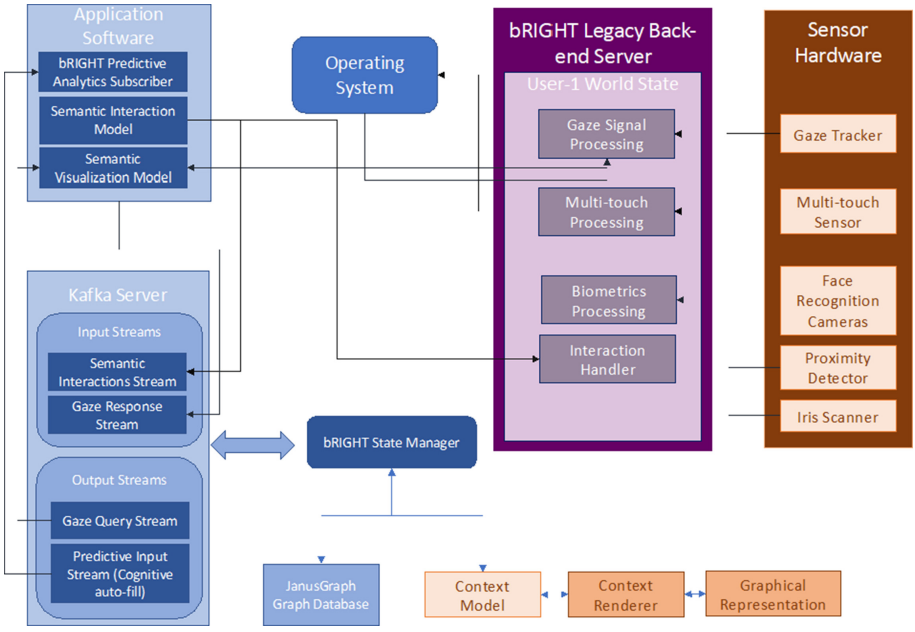

Fig. 9. The bRIGHT architectural overview depicting major system components.

Architecture of bRIGHT. The major system components of the bRIGHT architecture are shown in Fig. 9. In the following sections we describe the roles of components and how they integrate as a cohesive HCI framework that facilitates our research. This is a

technical viewpoint of context modeling that facilitates our user-centered research. Our methodological viewpoint will be described in a future publication.

Extending the Application Model. The bRIGHT system is designed as a research framework and concept design for a future workstation. It allows us to investigate application paradigms such as object-oriented programming, which supports capture of accurate user context. bRIGHT has extensions to a typical application's business logic that include the bRIGHT analytics subscriber module (BASM), semantic interaction model (SIM), and the semantic visualization model (SVM) for each application. Further information about the application modeling enhancements in bRIGHT can be found in [19, 20]. The BASM is a component that connects to the bRIGHT analytical reasoner via the Kafka output stream from the bRIGHT state manager. The main purpose of BASM is to update the application's state with input from the bRIGHT analytics module. Such input could provide predictive data when an analytics module determines the user will need to match a required entry in a field in the application interface.

Knowledge Representation. The persistent data storage model for bRIGHT's state management uses a distributed, open-source, massively scalable graph database technology called JanusGraph. The scenarios in which graph databases outperform traditional relational databases are well documented. Given the nature of bRIGHT's knowledge representation, a graph representation is used as the primary form over which the analysis modules execute their reasoning. We use JanusGraph (part of the Apache TinkerPop[2] graph computing framework) and Gremlin[3] as the graph traversal machine and language for implementation of the connected component representing bRIGHT's world state.

The bRIGHT state manager (BSM) is the component responsible for connecting to the JanusGraph database back-end and populating the initial world state when the bRIGHT system powers up (see Fig. 9 above). In addition, the BSM subscribes to all of the input and output streams from the Kafka server, and based on the data flow on these streams, builds and manages the context model. The bRIGHT analytics module connects to the BSM and uses updates from the BSM to run various run-time analyses of the graph representation to enable predictive input and task automation, etc. The results are sent back to the BSM as state updates and passed along to the corresponding components of the system via the Kafka output streams. The bRIGHT analytics module also uses graph analytics techniques to identify regions of the graph or relationships that may be of interest in studies related to long-term evolution of context. These features are experimental and in advanced prototype stages.

Context Model. The BSM creates the context model based on various graph queries that execute due to triggers received from the Kafka input streams. This could be a user-action-associated semantic interaction model instance (i.e., pressing the "Send Mail" button in a mail client creates a SIM instance associated with being streamed into the BSM via the Kafka input stream), or it could be a gaze context response from the application for a query generated by the bRIGHT back-end server. This might happen because the user's gaze fell on a part of the screen being tracked for this application. The

[2] https://tinkerpop.apache.org/.
[3] https://tinkerpop.apache.org/gremlin.html.

graph queries utilize a system that is built on open domain knowledge and encapsulates some common concepts required for state management. Example domain knowledge could be the format of an IP address and its relationship to the network mask. Common concepts useful in such a case could be that IP addresses are often associated with geolocations, which contain properties such as "Country", "State", and "Zip Code".

Since the entire system is engineered end-to-end with the ability to stream signals such as gaze and also interprets the operational semantics of such signals at the user level—i.e., instead of updating the context model with gaze at point $P(X_i, Y_j)$, we update it with higher-level concepts such as: The user looked at the "Subject line of an unopened email containing …"—in real-time speed. As such, there is no loss of information in terms of what interactions the user took in engaging the system, and the context model reflects almost all the information and concepts the user is currently interested in. It is not an exaggeration to claim that all the major design decisions in the bRIGHT system were made to maintain high accuracy of the context model.

The BSM maintains a journal of all updates applied to the context model; therefore, the evolution of the context model can be played forward and back in time when experiments are conducted. The Kafka stream data are also available, and we can fully replay any specific experiment or scenario, recreating the results from the signal streams upward. This allows us to run various graph analysis algorithms over the evolution of the context model and develop insights into long-term effects and patterns.

In cybersecurity demonstrations and experiments, we have seen the context models grow to contain thousands of entries and fairly complex connected components within the span of a few hours. Most of the content is from the gaze-tracking input, since cyber operators consume a lot of information both in terms of transaction rate and volume. It is not surprising that the context model of a cyber operator grows rapidly as a consequence of workflow. Cyber operators are constantly monitoring complex intrusion detection and malware detection software as well as routinely communicating with multiple team members to effectively collaborate. This complex and detailed context model also highlights why it is extremely difficult to track operator interests and anticipate their needs without such a rich and complex representation.

Kafka Streaming Server. The 3rd-generation bRIGHT system generates approximately 6 gigabits of sensor data every second. In addition, it creates intermediate sensor processing results such as blob detection output from the multi-touch surface, proximity detectors, face recognition status sensors, and so forth. bRIGHT needs to handle all of these streams of high sustained transfer-rate-oriented signals while performing complex context-oriented reasoning. As such, we chose to improve the scalability of our signal streaming capacity by integrating the Apache Foundation's Kafka streaming server into the bRIGHT framework[4].

The Kafka server has bRIGHT-friendly input streams (see Fig. 9, application status data and sensor data) and output streams (gaze queries being sent to applications, task automation, or predictive input results from the analytics module). Any future experiments conducted using this platform will be done by integrating software relevant to the

[4] https://kafka.apache.org.

experiment and connecting to a new Kafka input stream and output stream specific to that experiment context.

One of bRIGHT's core abilities is tracking the user's interest and responding to queries in the user's context model to execute application functionality in the near future. This "predictive input" is an extremely accurate form of autofill. When certain applications are fully integrated with bRIGHT, input parameters can be populated for an application feature and will auto-execute if there is enough evidence in the context model to suggest the user may do this in the near future. Task automation and predictive input are some of the outputs from the bRIGHT analytics module that are streamed to the application software using the Kafka output. Accurately tracking the context of the user allows proactive engagement that supports the user with needed information for input and task automation.

6 Future Work and Conclusions

A major focus for future work is to evaluate bRIGHT in user studies. While we have applied bRIGHT internally to various projects to experiment with task automation and cognitive autofill as means to adapt to user context, we have yet to conduct large and formal user studies.

Given the revolutionary gains in performance of AI-based systems and the rapid increase in deployment of autonomous systems in defense and disaster recovery, every aspect of human-machine interaction (HMI) is transforming rapidly. Understanding the user's context and leveraging it has always been significant in HMI, but this is now becoming one of the key areas for research and development because a much broader and more accurate definition of context will be needed in the future. We can expand our ability to capture user context by integrating better sensor systems, as evidenced in autonomous driving systems; however, to achieve truly transformational gains, we need advances in fundamental computing principals such as application modeling paradigms. These must be updated from vastly outdated models such as object-oriented programming, etc. As we have shown in our previous work [19, 20], expanding modeling so the application can respond to contextual queries about rendering context on screen and user interactions is important. We must provide a rich operational semantics level to create highly accurate context models of the user.

By improving the knowledge representation scheme in bRIGHT and adding support to embed high-level decision theory constructs such as "value maximization", "semantic framing", "attribute framing", "loss aversion", and "temporal discounting" among others, we will support identification and tracking of human decision-making based on context.

At present, the entire bRIGHT framework is built upon a very narrow definition of the user's context. Adding support for high-level concepts such as societal and environmental characteristics will increase the broader impact of this technology.

Our work in the last 8 years has allowed us to capture information and adapt the bRIGHT platform to the user's context in the short term. The ability to track long-term evolution of human experiences, interests, and values is useful. Indeed, rapid advances in AI and the predominance of autonomous systems will change the future.

We have demonstrated a need to extend basic computing principles such as application modeling paradigms to better support understanding the user's context. We extended existing applications by adding semantic interaction models and semantic visualization models that better describe the user-level operational semantics of the interactions and on-screen content. When the user engages with the system, we are able to identify, track, and adapt their context so that a response to change happens in near-real-time fashion. As such, we designed the entire bRIGHT framework end-to-end with these requirements in mind. To make revolutionary gains in capturing a broad swath of the user context (including societal and environmental aspects), user-centered design processes must follow a holistic approach that accounts for advances in sensor systems, software architectures, and application modeling. Consideration of short-term trends and long-term evolutionary patterns will also be important.

References

1. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: First Workshop on Mobile Computing Systems and Applications, pp. 85–90. IEEE (1994)
2. Moran, T.P. (ed.): Special issue on context in design. Hum.-Comput. Interact. **9**, 1–149 (1994)
3. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A unifying reference framework for multi-target user interfaces. Interact. Comput. **15**(3), 289–308 (2003)
4. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. **2**(4), 263–277 (2007)
5. Ghiani, G., Manca, M., Paternò, F., Santoro, C.: Personalization of context-dependent applications through trigger-action rules. ACM Trans. Comput.-Hum. Interact. **24**(2), 1–33 (2017)
6. Matthews, T., Whittaker, S., Moran, T., Yuen, S.: Collaboration personas: a new approach to designing workplace collaboration tools. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2247–2256 (2011)
7. Wang, R., Larusdottir, M., Cajander, Å.: Describing digital work environment through contextual personas. IFIG WG 13.2 Workhop at INTERACT 2021 (2021)
8. Thew, S., Sutcliffe, A.: Value-based requirements engineering: method and experience. Require. Eng. **23**(4), 443–464 (2017). https://doi.org/10.1007/s00766-017-0273-y
9. Sutcliffe, A.: Conflicting requirements and design trade-offs. In: IFIP WG 13.2 + 13.5 Workshop on Dealing with Conflicting User Interface Properties in User-Centered Development Processes. Mumbai (2017)
10. Johansen, P.S., Jacobsen, R.M., Bysted, L.B.L., Skov, M.B., Papachristos, E.: Designing a machine learning-based system to augment the work processes of medical secretaries. In: Loizides, F., Winckler, M., Chatterjee, U., Abdelnour-Nocera, J., Parmaxi, A. (eds.) Human Computer Interaction and Emerging Technologies: Adjunct Proceedings from the INTERACT 2019 Workshops, pp. 191–196. Cardiff University Press, Cardiff (2020)
11. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2013)
12. Akiki, P.A., Bandara, A.K., Yu, Y.: RBUIS: simplifying enterprise application user interfaces through engineering role-based adaptive behavior. In: Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. pp. 3–12 (2013)
13. Gajos, K., Weld, D.S.: SUPPLE: automatically generating user interfaces. In: Proceedings of the 9th International Conference on Intelligent User Interfaces, pp. 93–100 (2004)

14. Yigitbas, E., Jovanovikj, I., Biermeier, K., Sauer, S., Engels, G.: Integrated model-driven development of self-adaptive user interfaces. Softw. Syst. Model. **19**(5), 1057–1081 (2020). https://doi.org/10.1007/s10270-020-00777-7
15. Skillen, K.-L., Chen, L., Nugent, C.D., Donnelly, M.P., Burns, W., Solheim, I.: Ontological user profile modeling for context-aware application personalization. In: Bravo, J., López-de-Ipiña, D., Moya, F. (eds.) UCAmI 2012. LNCS, vol. 7656, pp. 261–268. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35377-2_36
16. Chen, H., Finin, T.: An ontology for a context aware pervasive computing environment. In: IJCAI Workshop on Ontologies and Distributed Systems, Acapulco (2005)
17. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using OWL. In: IEEE Annual Conference on Pervasive Computing and Communications Workshops, Proceedings of the Second, pp. 18–22. IEEE (2004)
18. Porras, P.A., Senanayake, R., Kaehler, J.: Revolutionizing the visual design of capture the flag (CTF) competitions. In: Proceedings of the 21st International Conference on Human-Computer Interaction HCI'19. Orlando, Florida, USA 26–31 July 2019
19. Senanayake, R., Denker, G., Lincoln, P.: bRIGHT – workstations of the future and leveraging contextual models. In: Yamamoto, S., Mori, H. (eds.) HIMI 2018. LNCS, vol. 10904, pp. 346–357. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92043-6_29
20. Senanayake, R., Denker, G.: Workstations of the future for transformational gains in solving complex problems. In: Kurosu, M. (ed.) HCII 2019. LNCS, vol. 11568, pp. 476–488. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22636-7_36