Hybrid Microgenetic Analysis: Using Activity Codebooks to Identify and Characterize Creative Process

Cesar Torres, Matthew Jörke, Emily Hill, Eric Paulos
Electrical Engineering and Computer Sciences
University of California, Berkeley
[cearto, emilyhill, mjoerke, paulos]@berkeley.edu

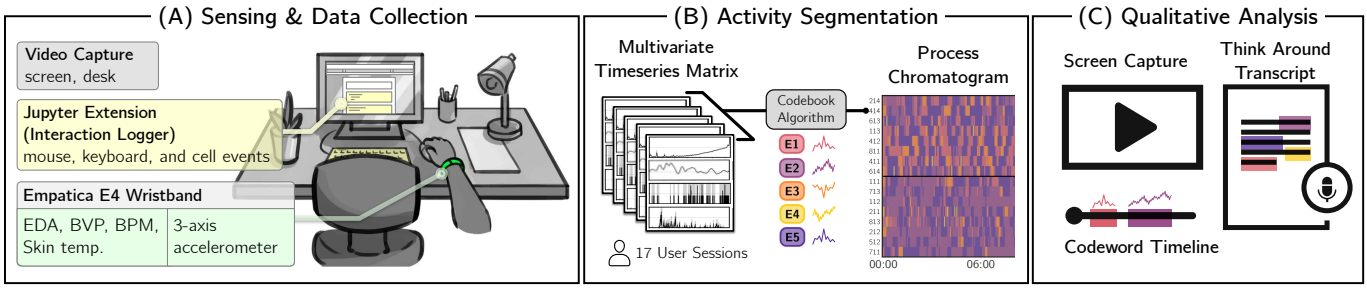

Figure 1. Hybrid Microgenetic Analysis: (A) The environment and user are instrumented to record creative activity; (B) Time-series data is processed to generate a codebook representing distinct subsequences of activity; the codebook is visualized as a process chromatogram; (C) The chromatogram is used to guide qualitative analysis, highlighting periods of unique activity between or within users.

ABSTRACT

Tacit knowledge is a type of knowledge often existing in one's subconscious or embodied in muscle memory. Such knowledge is pervasive in creative practices yet remains difficult to observe or codify. To better understand tacit knowledge, we introduce a design method that leverages time-series data (interaction logs, physical sensor, and biosignal data) to isolate unique actions and behaviors between groups of users. This method is enacted in Eluent, a tool that distills hundreds of hours of dense activity data using an activity segmentation algorithm into a *codebook* — a set of distinct, characteristic sequences that comprise an activity. The results are made visually parsable in a representation we term *process chromatograms* that aid with 1) highlighting distinct periods of activity in creative sessions, 2) identifying distinct groups of users, and 3) characterizing periods of activity. We demonstrate the value of our method through a study of tacit process within computational notebooks and discuss ways process chromatograms can act as a knowledge mining technique, an evaluation metric, and a design-informing visualization.

CCS Concepts

•**Human-centered computing** → **User models;** *Ethnographic studies; Dendrograms;*

Author Keywords

Tacit knowledge; digital ethnography; activity segementation

C&C '19 June 23–26, 2019, San Diego, CA, USA

ACM ISBN 978-1-4503-5917-7/19/06.
DOI: https://doi.org/10.1145/3325480.3325498

INTRODUCTION

A creative process, or how one moves, trains, conditions, and paces the mind and body, allows for practitioners to overcome difficulties and sustain practice — essential elements to developing skill, building efficacy, and fostering creativity [21]. Process is a critical component of any creative practice yet remains a tacit skill[1] especially within STEM fields that prioritize theoretical knowledge. Creative technologies have the potential to shape this creative process, supporting the available skills of the practitioner and mediating a creative practice.

Several techniques for studying tacit knowledge are impacted by two major factors: (1) the difficulty in vocalizing or observing tacit knowledge, and (2) the expense of collecting, processing, and analyzing ethnographic data. In this work, we introduce a design method called Hybrid Microgenetic Analysis (HMA) which computationally compiles a descriptor of tacit activity called a *process chromatogram* that highlights and identifies unique users or periods of activity to seed ethnographic and design inquiry. The method is not fully automated and instead acts as an aid for designers analyzing sensor data. We are especially interested in understanding and documenting a level of detail that occurs over a few seconds (micro) within a creative practice, such as the pressure exerted by a calligrapher's pen, the "emotional rollercoaster" experienced by programmers, or the routine evolution of a hundred piped roses by a cake decorator. This detail is provided by sensor (e.g., biosensors, accelerometers) and activity logs in the creative workspace. Following a microgenetic design, data is captured for multiple practitioners (or the same practitioner, many times) undergoing the same task to observe differences in process or the evolution of skill.

[1]Tacit knowledge is a form of procedural knowledge of how to perform activities [18]. Polanyi [34] described tacit knowledge as "things that we know but cannot tell" often existing in the subconscious.

To synthesize the thousands of hours of collected data, we employ a *codebook* approach which takes the full corpus of activity data and distills a codebook of k codewords that represent distinctive actions or behaviors. This codebook is then used to discretize time series data and characterize periods of activity. Codebook approaches have shown promising results in identifying characteristic features of emotion, motion, and cognition activity from electrooculography, accelerometer, and biosensors [24, 40]. We take the additional step of visualizing codebooks as *process chromatograms*. Similar to the role of liquid chromatography in identifying components of a substance, process chromatograms help identify components of an activity and make patterns more salient. These patterns are used to guide researchers in identifying individuals and periods of activity to compare. For instance, a codeword representing an absence of motion (from accelerometer data) can be used to characterize the difference in action versus thinking between novice and expert practitioners. We enacted this design method in Eluent[2], a tool that processes sensor data, runs an activity segmentation algorithm to produce codebooks and chromatograms, and provides a qualitative analysis interface for navigating sensor activity data against traditional ethnographic data (e.g., interviews, questionnaires, field notes)(Figure 1).

In this paper, we first position our design method in the space of research methodologies that study tacit process. Though HMA can be applied broadly for understanding tacit processes across domains, this paper instantiates these ideas in the context of a case study examining programming style and workflow within computational notebooks[3]. We showcase the role of a process chromatogram in providing additional ethnographic description in a microgenetic analysis of 17 participants carrying out a programming task, and 2) the ability to characterize emotion and interaction behaviors. We conclude with a discussion of ethical issues arising from hybrid microgenetic analysis and the role process chromatograms have as a knowledge mining and evaluation technique, a digital pedagogical tool, and as a design-informing visualization.

RELATED WORK

In this section, we review studies of tacit process, focusing on research methods and systems for analyzing, recognizing, modeling, and visualizing creative activity.

Studying tacit process

Several methods aim to understand and describe tacit process operate at different levels of granularity from careful observation of seconds of activity to multi-year longitudinal studies. Within cognitive science, microgenetic [42] and nanogenetic methods [28] observe "the moment-by-moment change" over time within repeated tasks (e.g., photocopying [41], spreadsheet calculations [22]). These methods aim to track and qualify the development of skill over time and understand the "person/environment dyad" [41]. While providing rich qualitative insights about learning, microgenetic analysis requires study environments to be carefully instrumented to isolate task behaviors, and, even then, require hours of hand coding.

Computational and sensor-supported ethnography [50], or the use of unobtrusive sensors or other computational methods for ethnographic inquiry allows for a richer description of activity. Our work falls within this family of methods, specifically lensed towards creative practice. In these methods, sensors are used to "shadow" the user, obtaining difficult to obtain data from in-person interaction; however, the data offers only a partial view of a situated perspective that considers social interactions and cultural histories [33] and is limited by privacy concerns [7]. Low-cost sensing infrastructure is continuing to develop [7, 44] to capture richer information about user interactions, including unique perspectives from non-human actors [14, 5]. Processing and incorporating sensor data into analysis continues to remain a challenge especially with the range of data encoding formats and volume of information.

Tacitness is a property of the knower: it is easily articulated by one person but may be very difficult to externalize by another [45]. Design ethnographic methods like contextual inquiry [19] rely heavily on the knower's ability to articulate and the observer's ability to discern what to examine. In contrast to these approaches, we used sensed and captured information to examine activity that would be otherwise imperceivable. Within creative practices, ethnographic methods seek to foreground difficult to observe activity, leveraging expert learners to experience and document mental models [49], or altering the saliency of stimuli (e.g., blindfolding to highlight haptic experiences [17]). Methods have been proposed for synthesizing observed creative activity with cognitive models, such as enactivism [9], to formulate a continuous representation of activity. Within programming (our case study domain), activity has been observed from interaction logs [31], version control histories [38], and biosensors [30] in practices such as entry-level programming [23] and debugging [25]. Our proposed design method is a form of microgenetic analysis that incorporates a diverse set of time series data to capture a more holistic view of process while employing traditional ethnographic data to condition our interpretation of computational data. Using unsupervised methods, the data is coded but unlabeled, sitting on the edge of ambiguity yet providing quantitative hints to guide a qualitative analysis of data.

Programming styles

Process emerges from a variety of contributing factors. Environmental factors shape the ease in which materials and tools can be accessed. The proliferation of search interfaces for programming snippets (e.g., StackOverflow) encourage practitioners to develop foraging behaviors [4]. Social practices around research have influenced key behaviors found in computational notebooks: tracking provenance, reusing code, enabling replication, and presenting results [36]. Discussions surrounding epistemological pluralism have recognized that a spectrum of approaches to programming exists, the most prominent of which contrasts the *hard* approach, characterized by logical, plan-oriented actions, with a *soft* approach, distinguished by a desire to work closely with the objects rather than abstraction [46]. This distinction has influenced

[2]In liquid chromatography, the eluent refers to the solvent that breaks down and transports the substance under analysis.

[3]Computational notebooks refer to a class of integrated development environments (IDEs) that facilitate journaling and documentation while programming.

emergent practices such as block-based flow programming (e.g. Scratch [35]), code bending [2], and embodied tacit skills like code smelling [11, 36]. These styles align with a larger conversation around making, extending beyond programming to other creative domains. Under the *hylomorphic* model, practitioners impose forms internal to the mind upon a material world; in contrast, under the *morphogenetic* model, a practitioner "joins forces" with active materials to react, synthesize, and anticipate new forms [20]. This morphogenetic process is the often unconscious reality of many programmers' natural practice, such as in the act of debugging. HMA acts as a method to make more salient how process emerges in practice.

Systems and models of creative activity

A streamlined system architecture is needed to coordinate, synchronize, and organize diverse sensing implements. With growing ubiquitous IoT sensing technologies, Wilkinson et al. demonstrated the value of a common time window format for sensors in a cooking production set [48]. Notably, the architecture was not geared towards activity recognition, but instead held a model of usage that could be used by editors to suggest where the focus of production should be. Similarly, our approach does not aim to classify activity but provide information about patterns and trends to inform design inquiry.

Several temporal data mining techniques exist for extracting patterns of actions and behaviors [12]. Markov chains and n-grams were used to construct probabilistic models of activity [50] at different granularities of patterns, segments, sequences, and events [26]. A clustering approach identified similar styles in a corpus of student programming submissions [6]. Our approach leverages codebooks, a technique that represents activity as a distribution of characteristic subsequences [40], often used to detect good features for machine learning routines [13] and making sense of difficult-to-interpret data, like electrooculographs [24].

Visualizing multivariate time-series data has a long history within the data visualization communities [1]. Pattern-highlighting techniques include brush and linking across data sources [26], sketch-based interactions for exploring time-series data [27], and patination techniques usage patterns rendered as heatmaps [31]. Most similar to our approach is MotionExplorer [3], an interactive dendrogram visualization tool applied to human motion data that generates cluster glyphs to represent both higher-level behaviors and discretize time-series data. We build on this approach leveraging codebooks extracted from dendrograms to discretize sensor streams into visually-parsable descriptors of activity.

METHOD AND TOOL DESIGN

We describe a 3-stage method termed Hybrid Microgenetic Analysis (HMA) for studying and documenting process (Figure 1). The method, operationalized in Eluent, involves 1) configuring the environment to sense relevant activity and collect data from a microgenetic study, 2) distilling a codebook for producing a visual descriptor of activity, and 3) using the descriptor to guide a qualitative analysis of process.[4] The method design is motivated from Polanyi's [34] definition of tacit knowledge: tacit knowledge is a relationship between the *proximal*, referring to sensations and perceptions of the human body, in response to the *distal*, or the object in focus. Within crafting theory, this relationship guides the maker in the *wayfaring* process [39], leveraging the responses from the material to inform future actions and intentions. Under this definition, understanding tacit process relies heavily on information received from a practitioner's sensory system, as well as the objects in focus within the environment.

Sensing and Data Collection

For creative practices, sensing should capture three types of information: 1) information about the body and the activation of sensory receptors (proximal); 2) information about the environment (distal); and 3) information of the object in focus. Careful consideration needs to be taken in sensor selection and placement to reduce observer effects and not influence practitioners' natural behavior. To capture this type of information for our case study, we created a work environment composed of a PC with a 19" monitor running a Jupyter[5] Notebook, mouse and keyboard, and scratch paper (Figure 1A). The following data collection mechanisms were introduced:

- **Physical Body:** Biosignals, such as electrodermal activity (EDA) and heart rate (HR) are recorded using an Empatica E4 Wristband[6]. The wristband has been used in emotion and sentiment analysis [29] and is capable of logging biosignal for realtime or post-hoc analysis. We used convex optimization [16] to extract the EDA phasic component (short-term physiological responses to event stimulus).
- **Environment:** The open-source Jupyter package provides a method of tracking and detecting higher-order programming actions. We created a custom Jupyter Notebook extension that logs user actions including code, key, and mouse events. A screen capture documents interactions with external resources and webpages. An iPad records video of any off-screen activities such as writing on paper.
- **Attention:** After a recording a creative session, participants reviewed the captured video and reflected on their thought process and emotions in a retrospective think aloud; this technique has been shown to be less disruptive and offers similar information as concurrent think aloud [47].

In our case study, this setup was rated 5.3 ± 1.2 on a 7-pt Likert as eliciting natural coding behaviors. Although participants were aware that the iPad was only capturing desk activity, they reported more pronounced observer effects. Each data collection mechanism was configured to annotate each data point with a real-time clock (RTC) value or with a start time and sampling frequency. This common format mitigated major issues with synchronization (starting sensors/sampling) and failure (a malfunctioning sensing unit) while providing avenues for extendability (incorporating new sensors). For each session, acquired sensor data was stored as a multivariate time-series (MTS) matrix. Non-uniformly sampled features (e.g., Jupyter notebook events) were interpolated to match the sampling frequency of synchronous data.

[4] Eluent has been made open-source and available here: `https://github.com/Hybrid-Ecologies/eluent`.

[5] `http://jupyter.org/`

[6] `https://www.empatica.com/en-eu/research/e4/`

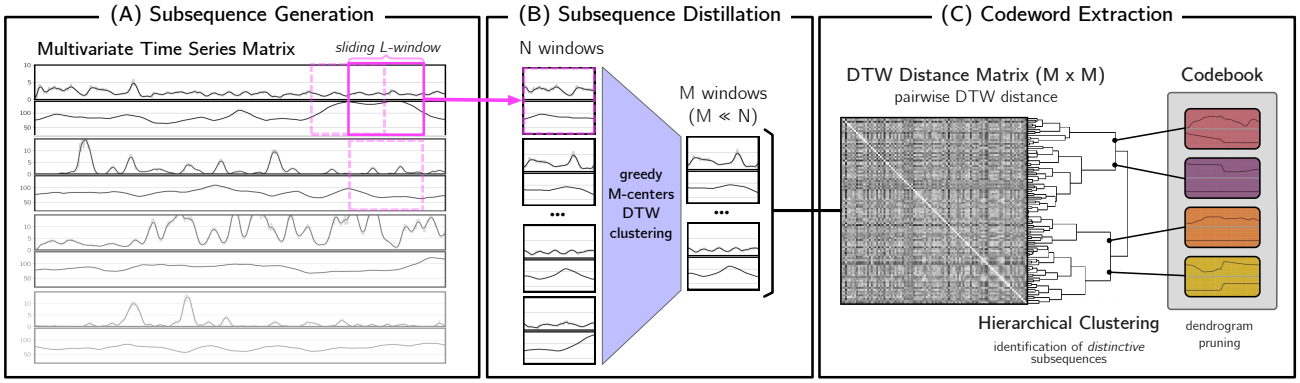

Figure 2. The activity segmentation routine. (A) Raw activity data is cleaned, aligned, and compiled into a multivariate time series (MTS matrix). (B) N subsequences of length L are collected from the MTS matrix and are sampled using greedy M-center clustering to identify $M \ll N$ maximally distinct samples. (C) The distance between each pair of samples is computed using DTW and used to construct a dendrogram (full-linkage hiearchical clustering). (D) The resulting dendrogram is pruned to extract a codebook of k codewords.

Codebook Activity Segmentation Algorithm

To compare and contrast behavior across different practitioners, a host of computational multivariate time-series analyses techniques can be used to identify larger-order behaviors and actions [12]. We developed an activity segmentation algorithm depicted in Figure 2 that leverages a codebook approach for discretizing time-series datasets. Our algorithm is adapted from Oates et al. [32], which identifies distinctive subsequences, or codewords, of size L between sets of multivariate time-series data. By introducing a greedy sampling routine, our adaptation is computationally tractable on large datasets (23K data points in 150 s on consumer systems [8GB RAM]). The algorithm requires a user-specified value for k, the number of words to extract (a data-dependent value), which then carries out three distinct steps: (1) *distilling* the full MTS matrix into a smaller set of characteristic sequences, (2) *extracting* k codewords from the characteristic set using hierarchical clustering, and (3) *applying* the resulting codebook to the original MTS matrix. By assigning each codeword a distinctive color, the final application steps transforms the MTS matrix into a *process chromatogram* (Figure 3).

Distilling Subsequences

The distillation step aims to reduce the size of the MTS dataset (on the order of gigabytes) before the computationally-expensive codeword extraction step. After cleaning, time-aligning, and normalizing the full MTS matrix, the matrix is partitioned into windows of size L for a total of N windows.

Data cleaning and alignment. For each feature in the MTS matrix, outliers exceeding the 99th percentile are clamped in value. Multivariate subsequences of length L are then extracted using a moving window with 50% overlap. The length L is chosen based on the sampling frequency of the data and the temporal range of activity. For our case study, windows corresponding to 3-5s of activity were chosen. Smaller windows (less than 2s), encroaching on *nanogenetic* analysis, suffered from stability issues during hierarchical clustering. Windows with missing data points (from malfunctioning sensing units/synchronization) are culled.

Comparing sequences. Since the dataset is in non-euclidean space, we utilize a Dynamic Time Warping (DTW) implementation from the Python `tslearn` library to compare sequences. As a distance metric, DTW is commonly applied to time series data due to its robustness to phase shifts and dilations in time. In multivariate settings, the DTW distance can either be computed by jointly measuring distance in multi-dimensional feature space or by summing univariate DTW distances across each dimension—we employ the former approach to reflect the tight temporal coupling of our features (e.g. the X/Y/Z components in accelerometer data should *not* be temporally decoupled). *Each L-window is independently z-normalized before computing the DTW distance, which preserves local shape and improves robustness to outlier values.*

Sampling the Window Corpus. The corpus of N subsequences is sampled using an adaptive greedy centers algorithm (DTW distance metric), yielding a sampled set of $M \ll N$ sequences. This method is preferable to random sampling because it produces maximally distinct samples—random sampling is likely to miss rare but distinctive subsequences. At step i of the greedy-centers algorithm, the i+1-th center is assigned to the data point with the maximum DTW distance from the i-th center. As opposed to choosing the number of clusters (M) in advance, the algorithm is adaptive to the dataset using a culling routine to terminate sampling: when a new center is identified, all subsequences that are below a cull threshold ε in distance to the i-th center are removed until all subsequences have been exhausted. We adjust the cull threshold to yield roughly $M = 1000$ centers.

Extracting Codewords

From the resulting sample of M subsequences, a set of characteristic sequences, or *codewords*, are extracted representing unique patterns of behaviors observed in the data. The number of codewords is left as a hyperparameter k. To extract codewords, we use agglomerative hierarchical clustering, an expensive operation that is expensive to run on the full set of N subsequences ($O(n^2)$ time, $O(n^2)$ memory [10]), hence the need for the previous greedy sampling routine. In agglomerative clustering, clusters are constructed from the bottom up, merging the closest two clusters at every step. We use a complete-linkage (i.e. farthest neighbor) clustering formulation, where the similarity of two clusters is defined as the distance between its most distant members.

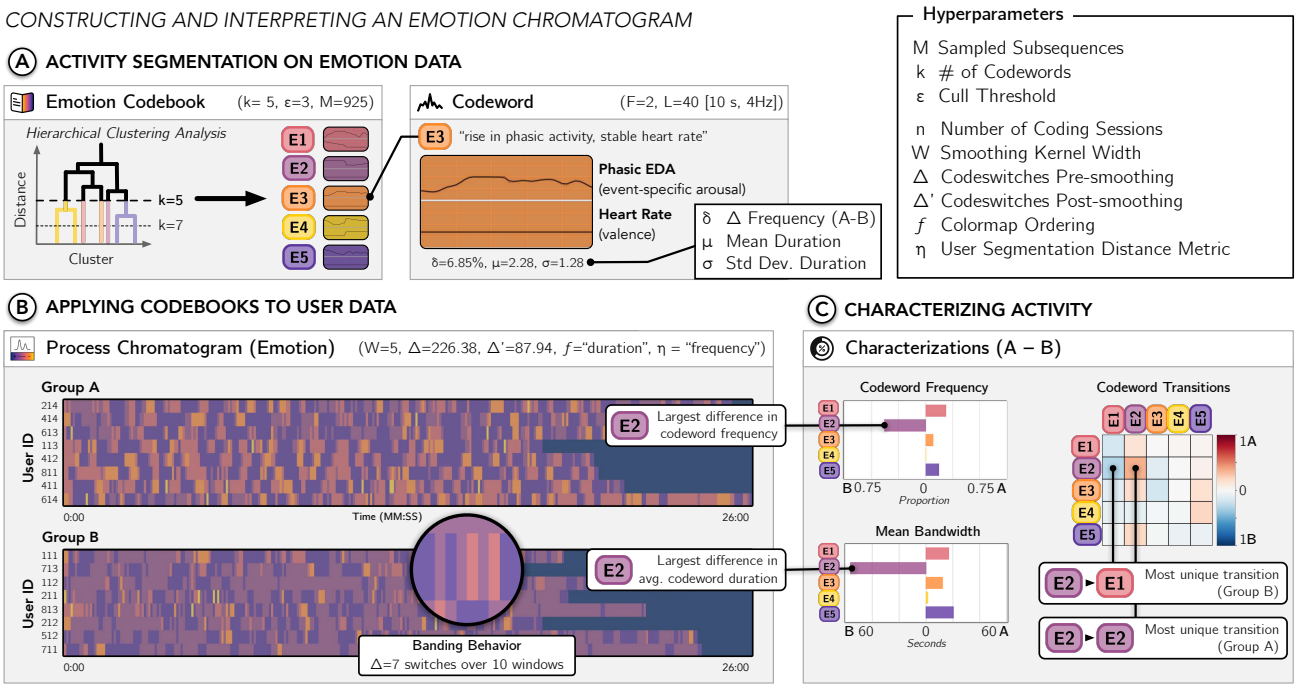

Figure 3. Interpreting a Process Chromatogram. Activity segmentation is run on a subset of features known to signal emotional cues. a) The most distinct windows of activity are identified as codewords and assembled into an emotion codebook; b) The codebook is applied to the activity data, assigning the closest codeword to each window. Each user's activity is clustered (by codeword frequency) to identify Group A and Group B; c) statistics are computed on codeword frequency, codeword duration, and transition probabilities to distinguish unique activity between Group A and B.

Hierarchical clustering has a visual representation called a *dendrogram*, a tree diagram that offers the advantage of caching expensive clustering computation in its structure. This allows the number of codewords k to be chosen *a posteriori* (Figure 3A). Clusters are collected into sets by pruning the dendrogram at the k-th level and collecting the leaves of each respective branch. Each codeword is computed by finding the clustroid (DTW distance metric). The final codewords are then collected in a set to form a *codebook*.

Applying Codebooks

Lastly, windows from the original user activity MTS matrix are assigned to the most similar codeword from the resulting codebook (DTW distance metric). By assigning each codeword a distinctive color, the original MTS matrix results in a visualization we term a *process chromatogram* (Figure 3B). Process chromatograms are stored as Timed Text Tracks (WebVTT), or caption files, for easy integration with web media.

Visual Tuning. Raw chromatograms tend to suffer from banding effects, characterized by rapid codeswitches which varies greatly between datasets. Banding undermines visual interpretability but can be alleviated with smoothing (Figure 3B). For window smoothing, we constructed a $T \times k$ soft assignment matrix for each user, where entry (i, j) contains the DTW distance of window i (in the original user MTS) to codeword j. For a kernel of size W, the window is reassigned to the codeword j with the minimum sum of distances across windows. We use the average number of codeswitches per user Δ to evaluate visual interpretability, targeting a Δ' (post-smoothing) that maintains fidelity to the data ($\approx$ 100 codeswitches). To improve visual readability, the colormap f maps the most salient (i.e. brightest) colors to codewords with the shortest duration; this allows for greater acuity of shorter codewords in the denser regions of the chromatogram.

Characterization. For each user, we extracted characteristics from the final process chromatogram (Figure 3C), including:

1. *Frequency Distribution*: the frequency of each codeword over a session.
2. *Bandwidth Distribution*: the distribution of codeword bandwidth across a session. A band is defined as a sequence of L-windows all assigned to the same codeword. The bandwidth is the length of a band.
3. *Transition Probability Distribution*: the transition probability from codeword i to j represented as a stochastic matrix.

User Segmentation. By comparing these characteristics between users or groups, we can identify which codewords characterize the unique differences in activity (Figure 3C). Using these characteristics, we can segment users by clustering them into groups (complete-linkage clustering, cosine distance metric, feature vector η). In our user segmentation, we used codeword frequency distribution as our feature vector; the generated chromatograms are sorted based on the clustering hierarchy, separated by the two highest-level clusters of users (denoted Group A and Group B, Figure 3B).

Qualitative Analysis Interface

After generating the chromatogram, the aim of this stage is to identify the referent of the codeword and ascribe meaning to the actions and behaviors described by the word. A web-based qualitative analysis interface (Figure 5) aids with navigating sensor activity data against traditional ethnographic data (e.g., interviews, questionnaires, field notes). The interface is flexible to a variety of data formats, adapting to traditional work-

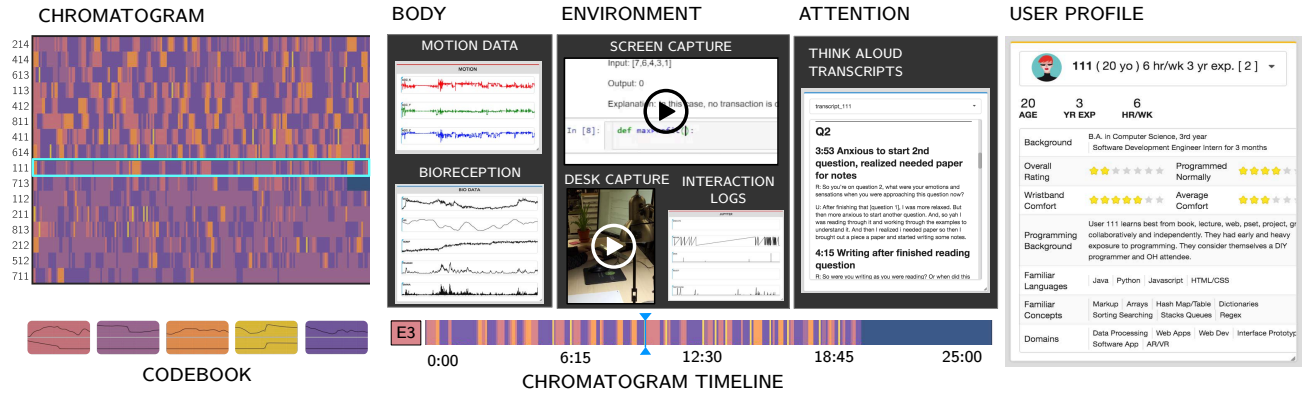

Figure 4. A web application links captured time-series data, think-aloud transcripts, and synthesizes a background profile from the background questionairre. All elements are brushed-and-linked and configurable to display raw sensor values or applied computed codebooks.

flows leveraging the file system to extract all files that are stored in a system folder, organized by participant. Multiple user-configurable windows render each of these files accordingly, providing a media player for video/audio files, plots for time series data and chromatograms, tables for questionnaires, and editable textareas for field notes and transcripts. Data sources with a time reference are brush-and-linked for easy reference, allowing users to use any source as a timeline. Using the chromatogram VTT caption files, we display the active codeword whenever the playback position is changed. This allows greater flexibility in leveraging HTML5 video features such as changing playback speed and looping. Chromatogram timelines can be used to toggle visible codewords.

CASE STUDY - JUPYTER NOTEBOOKS

To demonstrate the use of Hybrid Microgenetic Analysis, we conducted a study examining programming style and workflow within computational notebooks. These notebooks represent a shift away from traditional programming IDEs that produce artifacts with little information about a practitioner's process towards documentating process and intermediate results during the act of programming. We describe a data collection study and an HMA of emerging practices among users from a variety of backgrounds that use Jupyter notebooks.

Study Design

Participants. Participants were recruited from department mailing lists in Design, Engineering, and Computer Science. To reduce novelty effects, we only recruited participants with prior Jupyter iPython experience. A total of 17 participants took part in the study (8 female, 9 male, average age 22 $\pm$ 6 years). Participants completed a background questionnaire on education, programming experience, and affinity to the CS community. Educational backgrounds included 4 graduates, 12 undergraduates, and 1 full-time software engineer; 75% reported prior industry experience (avg. 5 years). Participants came from computer science, electrical engineering, data science, cognitive science, or mechanical engineering disciplines. Participants reported spending an average of 12 ± 10 hours programming per week; 70% had taken more than two university computer science courses. All but two participants identified as members of the CS community.

Study Protocol. Each participant was tasked with completing four programming interview questions sourced from LeetCode[7], presented in a single iPython notebook. The four tasks were chosen to represent a diversity of programming concepts and difficulties, solvable in any order: `T1`: a print-loop task, `T2`: a dynamic programming task, `T3`: a stable-sort task, and `T4`: a graph-traversal task. While we recognize LeetCode questions have many constraints, we chose questions that allowed considerable latitude in problem-solving approaches with the additional benefit of an established solution framework.

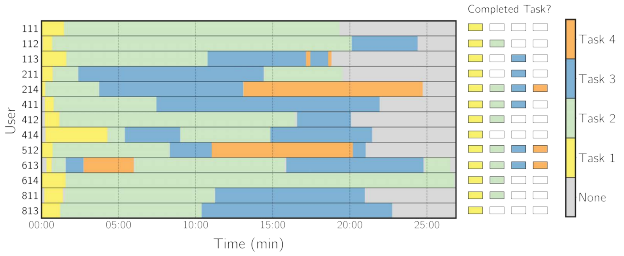

Figure 5. Time period spent on each task by participant. Only 214, 512, and 613 completed all four tasks successfully in the alloted time.

As described in Figure 1A, data was recorded from our Jupyter plugin, the E4 biosignal wristband, screen capture, and an iPad video recording the workspace. All data collection mechanisms were presented to the participant. Before each session, 5 minutes of E4 wristband were collected to establish a baseline. The other data capture mechanisms were initiated once the participant opened the Jupyter notebook. In pilot studies we observed participants experiencing anxiety when approaching the task's time limit. So as to capture a more natural programming session, we informed participants that a 40-minute time limit existed; however, they were prompted to stop earlier when less than 15 minutes remained in the study session. Participants then reviewed the screen capture recording of their session and narrated their programming strategies, emotions experienced, and process in a retrospective think aloud.

Data

The resulting MTS matrix consisted of 810K data points, 11.8 hours of video, 4 hours of transcripts which covered 5.9 hours of programming over 17 participants (on average 21 minutes per session). Due to equipment malfunctions, 2 participants did not have a screen capture, 1 participant did not have biosignal data, and 1 participant did not have a think-aloud transcript.

[7] `https://leetcode.com/`

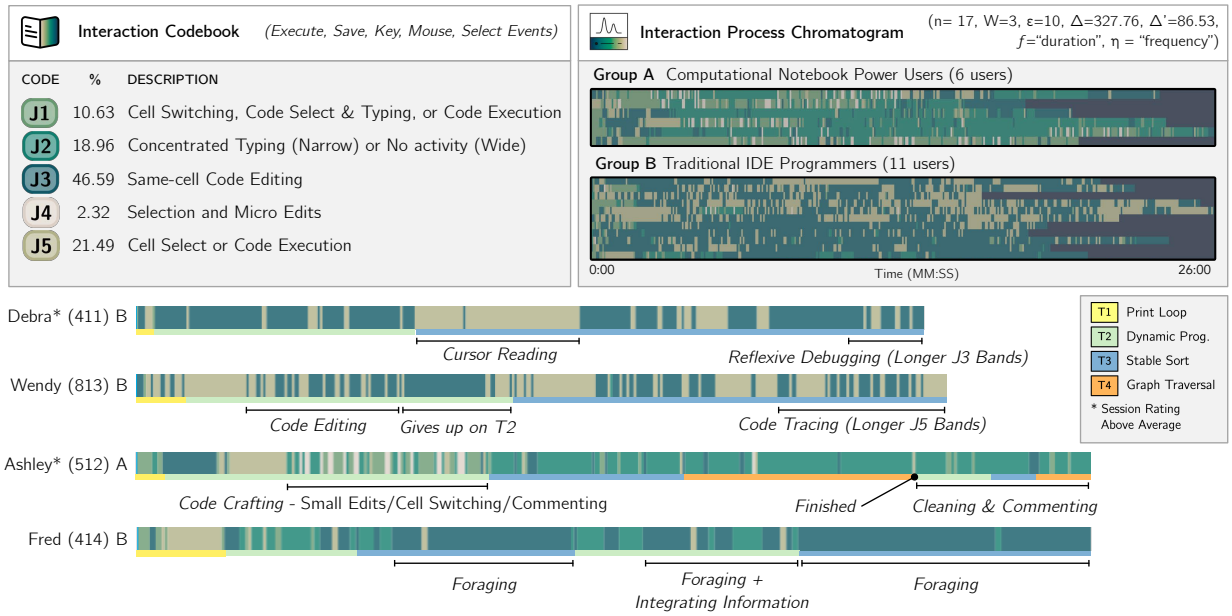

Figure 6. Interaction Behaviors (27K samples, 5 codewords)

For the respective analyses, these data points have been removed. Codebooks were computed over the full dataset on a subset of features: 1) an *emotion codebook*, a combination of heartrate (an indicator of valence), and phasic electrodermal activity (associated with shifts in arousal from event-driven stimuli), and 2) an *interaction codebook*, composed of Jupyter interaction logs features including save, execute, select, mouse, and key chunk events (# of uninterrupted words typed). Average codebook compilation time was 3 minutes.[8]

Hybrid Microgenetic Analysis

Process. Across two paper authors, activity data was inductively coded guided by the emotion and interaction chromatograms.[9] The coding phase was facilitated through a collaborative spreadsheet where memos were created for activity surrounding each task and codeword across participants. For example, memos for J2 describe 1) its expression with respect to task phase (expressed during the latter part of **T1-4**), 2) duration and composition of bands (typically seen in 1-4 seconds bands), and 3) the observed behavior of the user (activated around concentrated typing events). While the analysis focused on features associated with the codebook under review, we gradually incorporated other features into the analysis due to their high correlation with the phenomena being reviewed. For example, patterns in heartrate (emotion) were correlated to J2 windows (interaction). Transcripts were used to condition our interpretation and refine our inductive codes; themes were then synthesized and assigned a semantic label. From user segmentation, participants were clustered into two groups (A and B), and memos were used to distinguish differences in each group's behaviors.

[8]The full compiled chromatograms are available as supplemental materials. Colormaps were chosen from the `cmocean` thermal and rain colormaps — perceptually uniform color scales that can be easily disambiguated between different codebooks. Colors are assigned based on code bandwidth; chromatograms are sorted by user segmentation using codeword frequency as the activity feature vector η.

[9]Pseudonyms are used to protect participant anonymity.

The Interaction Codebook

Using the analysis interface, we reviewed the periods of activity associated with each of 5 codewords in the interaction codebook for 17 users and collected descriptions of observed behaviors (Figure 6). The micro-behaviors identified by the codebook activity segmentation algorithm described when, how quickly, how much, and in what context typing took place: J2 expressed for long spurts of key events such as in the case of transcribing a mental algorithm or commenting code. Three other codewords distinguished micro-edits: J4 expressed during *Tweaking* where a couple of characters are changed; larger edits where distinguished based on whether they occurred within the same cell (J3) or within different cells (J1). J5 was expressed when cells were selected but no typing had occurred; this was found in behaviors such as code execution or in *Cursor Reading*, where a user selects, highlights, and tracks text within a cell as they read. Debugging behaviors like *Code Tracing*, were characterized by alternating bands of J5 and J3, where a user walks through the algorithm, highlighting text as they step through each statement and make same-cell edits. Conversely, Code Tracing had similar characteristics to *Reflexive Debugging*, the act of using information from the debug console such as error or print statements to update code; in the chromatogram, it is expressed by stronger bands of J3 with bands from J5 and J4. Because of z-normalization in our algorithm, lack of Jupyter activity was mapped to either Codewords J2 and J3, but can be distinguished by notably longer band widths and refer to activities like writing on scratch paper or *Foraging*, i.e., looking for information such as through StackOverflow.

The most fluent of the computational notebook users, finishing all four tasks without encountering significant difficult, was Ashley (#512), a four-time undergraduate teaching assistant for an introductory computer science course well-versed in the class of interview questions in the study task (activity depicted in Figure 6). When approaching **T2**, a dynamic programming

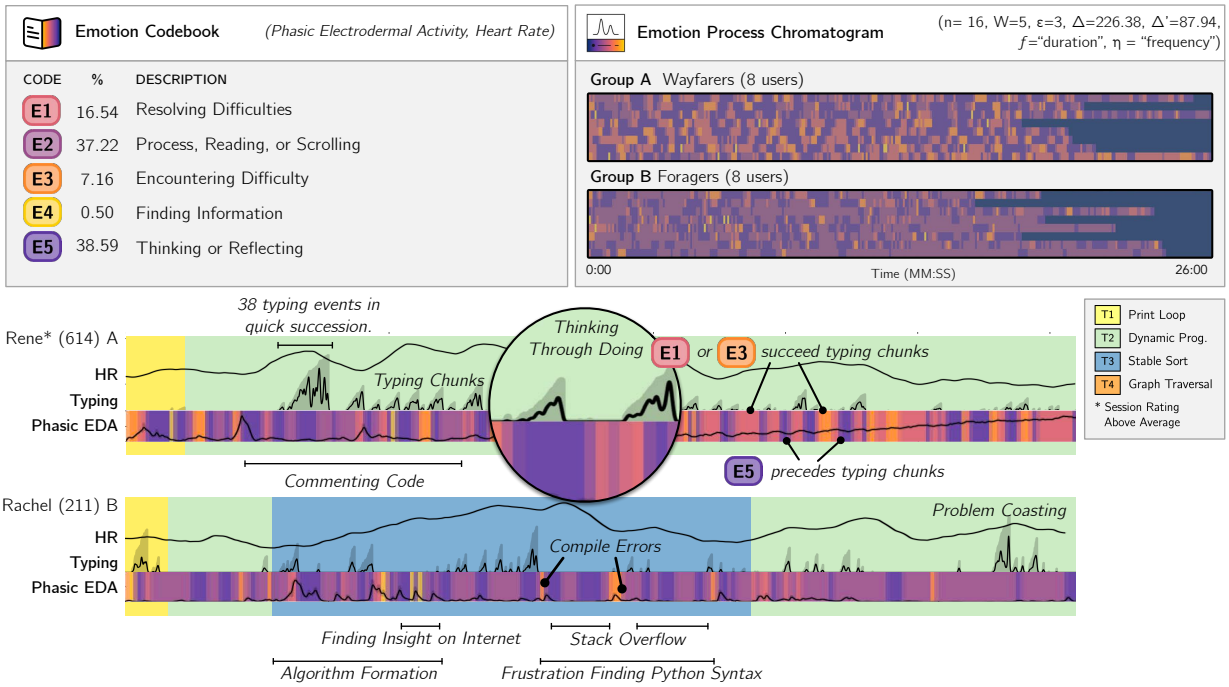

Figure 7. Emotion Behaviors (25K samples, 5 codewords)

task, she began by first writing a function definition and then proceeded directly to comment on the expected function behavior before any attempt of writing a mental algorithm or sketching pseudocode. She utilized the comments as a mechanism to "think about exactly what [she] wanted to write in the code instead of jumping in" and possibly "misinterpret the question." Ashley also noted that commenting added a sense of agency, where the action "builds [her] sense of commitment to a code that [she is] writing," and also inspires her to "make [the code] good if [she comments] it."

Ashley's algorithm writing process differed considerably from other participants, writing only a couple of lines before pausing to reflect and verify her work - a process we term *Code Crafting*. In contrast, others would engage in transcribing their entire mental or written algorithm without stopping to reflect. Notably, during this period of code crafting, Ashley would make micro-edits to the preceding lines in her code, before returning to add new lines. During the latter part of this process, she constructed test cases in a different cell followed by short periods of reflexive debugging, adding additional test cases until she was satisfied with the correctness of her solution.

The behavior of code crafting was made salient in the interaction chromatograph, consisting of rapid alternations of all codewords except J2 and J3 – the long-edit codewords. Ashley moved so quickly in both mentally compiling her code and proactively debugging her work that long periods of typing were rarely observed. The rest of her session is characterized by long bands of J2 which correspond to no Jupyter activity; these questions were perceived as more mentally workable by Ashley. As one of the only participants to finish all four tasks under the time limit, she additionally refined the document, doing everything from adding assert statements and preserving important elements of her process, to even restarting the Jupyter kernel to number the cells sequentially. Ashley's unique attention to detail and the aesthetics of her code was manifested through her active commenting and persistent code crafting throughout the coding session.

Overall, we clustered each user based on codeword frequency distribution, yielding two groups – A(6 users) and B(11 users). We annotate Group A as computational notebook power-users due to a higher frequency and higher average bandwidth of J1, J2,and J3 – codewords associated with actively using more than one cell and engaging with more concentrated typing. Notably, this group's interaction chromatographs show periods of code crafting. In contrast, Group B follows more traditional IDE workflows, keeping most coding activity to a single cell, writing pseudocode (on paper or in a cell), converting it into working code, and then engaging in reflexive debugging. Group A participants had higher-rated programming sessions (66% versus 54%); novice participants, distinguished by lack of professional programming experience, were more strongly associated with Group B (novices: 33% A, 64% B), suggesting that workflow can influence how programming sessions are perceived.

The Emotion Codebook

The analysis interface used Jupyter interaction logs, think aloud transcripts, and video recording to contextualize our interpretation of participants' emotions, interpreting biosignal on the Circumplex Model of Affect [37]. We reviewed activity associated with each of 5 codewords in the emotion codebook for 16 users (1 user had a malfunctioning biosignal recording). The codewords identified by the algorithm distinguished the type of phasic activity experienced: a short phasic response E5 corresponded to behaviors like thinking; longer, more sustained phasic activity E3 was more akin to encountering difficulties; a lack of phasic activity E2 was strongly corre-

lated to periods of reading, scrolling, and processing. Two codewords responded to changes in heartrate: for E1, falling heart rates correlated with participants resolving difficulties; for E4, rising heart rates were linked to individuals finding content especially when foraging (Figure 7).

René (#614), a graduate student in Computer Science with more than 20 years of industry experience in backend development, demonstrated a unique emotion chromatogram with the maximum number of 172 codeswitches (Δ) consisting of banding behaviors with E1, E3, E5; these bands are aligned to her *Typing Chunks*, a period of successive key events before any large pause. In contrast, Rachel(#211), a third-year undergraduate Mechanical Engineering student ($\Delta = 83$), had larger bands of activity with E2 primarily activated when she read and processed Stack Overflow content.

Both René and Rachel approached `T2`, a dynamic programming question with a solve-by-matrix approach, casting the problem to familiar 2D matrix manipulation that both had used in recent work. Both regularly encountered challenges of recalling Python syntax and programming conventions, yet Rachel abandoned the problem after 4 minutes, while René continued undeterred for another 24 minutes. René decided not to import pandas or numpy to ease matrix operations because she considered the dependency to be inconvenient for "someone else" who might be using this code. René faced syntax difficulties and Python memory and pointer management confusions, but she consciously decided not to consult online resources because she felt "it's unlikely to be exactly what [she] wants" and she "sort of know what [she] is doing."

Instead, René engaged in *Thinking Through Doing* behaviors, wrangling difficulties using *Probing Cells*, or peripheral cells dedicated to testing understand, probe, test, and validate syntax and code behaviors. This *Thinking Through Doing* strategy differed from other approaches such as foraging (Rachel) or *Reflection* (long periods of E2 and E5). During her probing process, René encountered many errors but was not annoyed, recognizing that "people get frustrated when they get errors" and that "random errors are inevitable", explaining her rationale for using probing cells. When interrupted at the end of the session, René refused to end the programming session and spent several more minutes before reaching a resolution. René's refusal to end the programming task session is a strong indication of René's resiliency. Although both encountered difficulties, we observed that René's phasic activity had large peaks concentrated in the initial solution formulation, not during the act of programming. From the think aloud, René indicated that she was in a state of flow. This suggests that the state of flow can act as *Creative Momentum*: if the body or mind is in motion, the emotional phenomena experienced during creative processes are less affected by stressors, such as encountering an error in programming.

Codewords expressed strongly for local phasic features, however, heartrate (a more global feature) was less present in the emotion chromatograph. When comparing heart rate against the time period spent on each task, we found that many participants expressed what we term *Problem Coasting* where each task exhibits "rollercoaster" heartrate forms with one or more peaks; valleys were observed during compile moments (Figure 7 Rachel). For René, `T2` had 8 peaks; in contrast, Rachel exhibited about 2 peaks per problem. These peaks also coincided with typing chunks. Although this phenomenon was not associated with successful or fast completion times, it does present itself as an indicator of stamina and resiliency.

Overall, we clustered users based on their emotion codeword frequency distribution, yielding two groups – A (8 users) and B (8 users). Group B had 66% more expression of E2 which corresponded to participants that engaged in more cognitive actions ("reading, thinking through the algorithm") such as Rachel; in contrast, Group A had more codeswitches of E1, E3, E4, like René. A paired-samples t-test was conducted to compare the number of codeswitches between groups. There was a significant difference in the scores for Group A (μ=100.4, σ=20.5) and Group B (μ=75.5, σ=19.1); t(7)=2.49, p=0.013. These findings suggest that Group A engages with more thinking-through-doing behaviors. For descriptive purposes, we label Group A as *Wayfarers*, working through and adapting to changes in the code content, and Group B as *Foragers*, searching for information.

DISCUSSION

In this work, we demonstrated how Hybrid Microgenetic Analysis could be used to both capture and characterize programmer's tacit process when using computational notebooks. We discuss the implications for computational ethnography, studies of tacit process, and IDE design.

Observing tacit process

Tacit process is difficult to observe; our study leveraged the open-source Jupyter notebook and E4 biosignal wristband to capture data that provided a more holistic profile of process. This profile made visible the relationship of emotion during the act of programming, revealing behaviors such as *Problem Coasting* that would otherwise be unobservable using traditional ethnographic approaches. It additionally provides a description of cognitive activity during processes like *Code Crafting* that distinguish it from *Reflexive Debugging*.

Although participants experienced some observer effects, they reported these effects abating over the course of the programming session. This suggests that conditioning analysis on latter parts of the session data could mitigate this bias. In addition, replicability is limited by equipment costs. We anticipate that sensing and activity logging technologies will develop to minimize observer effects (such as using mouse movement to detect stress [43]) that could remove the dependency on specialized hardware like the E4 wristband and be used in more naturalistic settings at scale. By using HMA, new insights, such as behaviors identified in our study, can be used to inform the development of specialized sensors or machine learning routines to accurately classify useful activity in realtime.

Scaling microgentic analysis

The volumes of data generated by microgenetic analysis or the amount of post-processing needed to extract important information limit the applicability of traditional microgenetic

analysis, especially in supporting quick, iterative design workflows. We demonstrated that our method is capable of processing 15 hours of data and 810K data points into process chromatograms in under 6 minutes; such chromatograms were used to guide the analysis of the data, identifying users and periods of activity to investigate.

The statistical power of amalgamating user data allowed some flexibility with data collection (e.g., corruption, faulty sensors). In our case, although biosignal data was lost for one session, the Jupyter interaction logs could still be used to inform the interaction codebook. Although our analysis focused on L =3-5 seconds behaviors with k =5 codewords, the activity segmentation algorithm allows for users to use hyperparameters k and L to select the level of analysis granularity. Lowering the number of codewords under analysis could be used to identify more robust classifications of behavior and inform feature engineering for machine learning algorithms. While the coding approach used in the case study was exhaustive (memos were created for each participant, task, and codeword), our process began with reviewing the activity of "exemplar" participants that represented centers from user clusters. We see potential for increasing the statistical power of the codebooks but using a principled sampling approach to conduct deeper qualitative analysis on such exemplar users. In Eluent, the degree of being able to label, annotate, or cross reference chromatograms was limited to brush-and-link interactions with traditional ethnographic data; future work can examine methods of facilitating direct annotations (e.g., as a word-scale visualization in transcripts [15]), searching for and labeling observed patterns, and automatically laying out labels (e.g., like those manually added in Figure 7 and 6) to create data-rich visualizations.

Mitigating bias in computational ethnography

The act of codifying tacit process allows for knowledge to be transferred through traditional information channels. However, widespread adoption and dissemination can lead to a single process becoming a standard, creating an epistemological monoculture that privileges those that align with the way of thinking it supports. HMA provides an avenue for comparing process among multiple groups of practitioners (versus asking only the "experts"). Through user segmentation, HMA revealed programming processes that break along boundaries other than level of expertise that can be used to inform a more plural codification of process. Being able to quantitatively characterize programming sessions allowed for individuals to be quickly compared and groups to emerge from the data which otherwise would not have been easy to identify. For instance, although professional programmers and graduate students exhibited more interactions with computational notebook cells, it was also the case that novice programmers from a MATLAB background employed a similar process. Such distinctions can be used to tailor instructional material to specific groups to develop best practices.

Activity is always partially observable which without context can easily be misinterpreted. As opposed to supervised methods that require hand labeling, our method uses unsupervised clustering to label time-series data. These labels are ambiguous but reflect a distinct activity observed in the data without imposing a meaning. We specifically incorporate an interpretation phase to build an understanding of phenomena observed in the process chromatogram conditioned against field notes, interview transcripts, and video recordings. Much of our setup was instrumented to sense useful information in the programming environment, but there exists a unique opportunity to have a mobile and streamable sensor toolkit for logging creative activity in both digital and physical settings. Maintaining privacy in such data collection environments requires a clear social contract between user and space, communicating how data is being used to inform resources. For collaborative practices such as peer programming, future work will explore how sensor data can be visualized and characterized as a dialectic to provide denser description of socio-technical interactions.

Informing the design of IDEs

A better understanding of tacit process has large educational implications, aiding in developing instruction that better facilitates skill acquisition and imbues resiliency in a practice. From our analysis, we observed heartrate patterns marking the beginning and end of solution attempts by participants in our study. Additionally, we noted the number of heartrate peaks over the course of a solution attempt as indicative of stamina and resiliency. Such patterns in heartrate can be used as an evaluation metric for computer science education that foregrounds resiliency as a marker of programming skill.

We also observed instances of creative momentum, or the continuous presence of external activity, like typing and cell switching, as mitigating high phasic electrodermal activity (associated with higher stress or cognitive load). This aligns with the phenomena of flow in creative psychology literature [8], a state of mind where a person becomes fully immersed in the activity at hand, and offers some evidence towards the value of flow in sustaining creative activity. Such behaviors that elicit creative momentum could be used as a design variable to aim for in creativity support tools. Such tools could maintain creative momentum by encouraging the user to engage in tedious, ritual tasks (e.g., refactoring code) when a heavy cognitive load (e.g., designing test cases) is expected.

CONCLUSION

By foregrounding *process*, this work invited awareness, critique, and discussion of a central element of creative development. This work introduced a method for describing tacit process within creative practices through Hybrid Microgenetic Analysis. We demonstrated the portability of the method in a digital practice – programming with computational notebooks. The analysis revealed distinctions between traditional IDE workflows and computational notebook workflows, heart rate patterns as indicators of resiliency and stamina, and the phenomena of creative momentum as regulating phasic electrodermal activity during the act of programming.

ACKNOWLEDGEMENTS

We thank Tina Taleb, Tony Zhao, and Sangyeon Lee for their assistance with the case study and the anonymous reviewers for their valuable feedback. This research was supported by a MSR Dissertation Grant and an NSF GRFP.

REFERENCES

[1] Wolfgang Aigner, Silvia Miksch, Wolfgang Müller, Heidrun Schumann, and Christian Tominski. 2007. Visualizing time-oriented dataâĂŤa systematic view. *Computers & Graphics* 31, 3 (2007), 401–409.

[2] Ilias Bergstrom and R. Beau Lotto. 2015. Code Bending: A New Creative Coding Practice. *Leonardo* 48, 1 (Feb. 2015), 25–31. DOI: http://dx.doi.org/10.1162/LEON_a_00934

[3] Jürgen Bernard, Nils Wilhelm, Björn Krüger, Thorsten May, Tobias Schreck, and Jörn Kohlhammer. 2013. Motionexplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2257–2266.

[4] Joel Brandt, Philip J. Guo, Joel Lewenstein, and Scott R. Klemmer. 2008. Opportunistic programming: how rapid ideation and prototyping occur in practice. In *Proceedings of the 4th International Workshop on End-user Software Engineering*. ACM Press, New York, NY, USA, 1–5. DOI: http://dx.doi.org/10.1145/1370847.1370848

[5] Wen-Wei Chang, Elisa Giaccardi, Lin-Lin Chen, and Rung-Huei Liang. 2017. "Interview with Things": A First-thing Perspective to Understand the Scooter's Everyday Socio-material Network in Taiwan. In *Proceedings of the 2017 Conference on Designing Interactive Systems (DIS '17)*. ACM, New York, NY, USA, 1001–1012. DOI: http://dx.doi.org/10.1145/3064663.3064717

[6] Rohan Roy Choudhury, HeZheng Yin, Joseph Moghadam, and Armando Fox. 2016. AutoStyle: Toward Coding Style Feedback At Scale. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW '16 Companion)*. ACM, New York, NY, USA, 21–24. DOI:http://dx.doi.org/10.1145/2818052.2874315

[7] Elizabeth F. Churchill, Ozzie Gooen, and David A. Shamma. 2010. Augmented ethnography: designing a sensor-based toolkit for ethnographers. In *Proceedings of the 2nNd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction*. ACM Press, New York, NY, USA, 416. DOI: http://dx.doi.org/10.1145/1952222.1952323

[8] Mihaly Csikszentmihalyi. 1997. Flow and the psychology of discovery and invention. *HarperPerennial, New York* 39 (1997).

[9] Nicholas Davis, Chih-Pin Hsiao, Kunwar Yashraj Singh, Brenda Lin, and Brian Magerko. 2017. Creative Sense-Making: Quantifying Interaction Dynamics in Co-Creation. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition (C&C '17)*. ACM, New York, NY, USA, 356–366. DOI: http://dx.doi.org/10.1145/3059454.3059478

[10] D. Defays. 1977. An efficient algorithm for a complete link method. *Comput. J.* 20, 4 (01 1977), 364–366. DOI: http://dx.doi.org/10.1093/comjnl/20.4.364

[11] Martin Fowler and Kent Beck. 2012. *Refactoring: Improving the Design of Existing Code*. Pearson Education.

[12] Ben D. Fulcher. 2017. Feature-based time-series analysis. *arXiv:1709.08055 [cs]* (Sept. 2017). http://arxiv.org/abs/1709.08055 arXiv: 1709.08055.

[13] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. 2010. Visual Word Ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 7 (July 2010), 1271–1283. DOI: http://dx.doi.org/10.1109/TPAMI.2009.132

[14] Elisa Giaccardi, Nazli Cila, Chris Speed, and Melissa Caldwell. 2016. Thing Ethnography: Doing Design Research with Non-Humans. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. ACM, New York, NY, USA, 377–387. DOI: http://dx.doi.org/10.1145/2901790.2901905

[15] Pascal Goffin, Wesley Willett, Jean-Daniel Fekete, and Petra Isenberg. 2014. Exploring the placement and design of word-scale visualizations. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2291–2300.

[16] A. Greco, G. Valenza, A. Lanata, E. P. Scilingo, and L. Citi. 2016. cvxEDA: A Convex Optimization Approach to Electrodermal Activity Processing. *IEEE Transactions on Biomedical Engineering* 63, 4 (April 2016), 797–804. DOI: http://dx.doi.org/10.1109/TBME.2015.2474131

[17] Camilla Groth, Maarit Mäkelä, and Pirita Seitamaa-Hakkarainen. 2015. Tactile augmentation: A multimethod for capturing experiential knowledge. *Craft Research* 6, 1 (2015), 57–81. DOI: http://dx.doi.org/doi:10.1386/crre.6.1.57_1

[18] Jennifer Hedlund, George B Forsythe, Joseph A Horvath, Wendy M Williams, Scott Snook, and Robert J Sternberg. 2003. Identifying and assessing tacit knowledge: Understanding the practical intelligence of military leaders. *The Leadership Quarterly* 14, 2 (2003), 117–140.

[19] Karen Holtzblatt and Sandra Jones. 1993. Contextual inquiry: A participatory technique for system design. *Participatory design: Principles and practices* (1993), 177–210.

[20] Tim Ingold. 2013. *Making: Anthropology, archaeology, art and architecture*. Routledge.

[21] James C Kaufman and Ronald A Beghetto. 2009. Beyond big and little: The four c model of creativity. *Review of general psychology* 13, 1 (2009), 1–12.

[22] Jong Kim and Frank Ritter. 2016. Microgenetic analysis of learning a task: its implications to cognitive modeling. In *Proceedings of the 14th International Conference on Cognitive Modeling (ICCM)*, David Reitter and Frank E. Ritter (Eds.). Penn State, University Park, PA, 21–26. http://acs.ist.psu.edu/iccm2016/proceedings/kim2016iccm.pdf

[23] Päivi Kinnunen and Beth Simon. 2011. CS Majors' Self-efficacy Perceptions in CS1: Results in Light of Social Cognitive Theory. In *Proceedings of the Seventh International Workshop on Computing Education Research (ICER '11)*. ACM, New York, NY, USA, 19–26. DOI:http://dx.doi.org/10.1145/2016911.2016917

[24] P. Lagodzinski, K. Shirahama, and M. Grzegorzek. 2018. Codebook-based electrooculography data analysis towards cognitive activity recognition. *Comput. Biol. Med.* 95 (April 2018), 277–287. DOI: http://dx.doi.org/10.1016/j.compbiomed.2017.10.026

[25] Colleen M. Lewis. 2012. The Importance of Students' Attention to Program State: A Case Study of Debugging Behavior. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research (ICER '12)*. ACM, New York, NY, USA, 127–134. DOI: http://dx.doi.org/10.1145/2361276.2361301

[26] Zhicheng Liu, Yang Wang, Mira Dontcheva, Matthew Hoffman, Seth Walker, and Alan Wilson. 2017. Patterns and Sequences: Interactive Exploration of Clickstreams to Understand Common Visitor Paths. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan. 2017), 321–330. DOI: http://dx.doi.org/10.1109/TVCG.2016.2598797

[27] Miro Mannino and Azza Abouzied. 2018. Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, 388:1–388:13. DOI: http://dx.doi.org/10.1145/3173574.3173962

[28] Taylor Martin, Ani Aghababyan, Jay Pfaffman, Jenna Olsen, Stephanie Baker, Philip Janisiewicz, Rachel Phillips, and Carmen Petrick Smith. 2013. Nanogenetic Learning Analytics: Illuminating Student Learning Pathways in an Online Fraction Game. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge (LAK '13)*. ACM, New York, NY, USA, 165–169. DOI: http://dx.doi.org/10.1145/2460296.2460328

[29] Cameron McCarthy, Nikhilesh Pradhan, Calum Redpath, and Andy Adler. 2016. Validation of the Empatica E4 wristband. In *Student Conference (ISC), 2016 IEEE EMBS International*. IEEE, 1–4.

[30] Sebastian C. Muller and Thomas Fritz. 2015. Stuck and Frustrated or in Flow and Happy: Sensing Developers' Emotions and Progress. IEEE, 688–699. DOI: http://dx.doi.org/10.1109/ICSE.2015.334

[31] Alok Mysore and Philip J. Guo. 2018. Porta: Profiling Software Tutorials Using Operating-System-Wide Activity Tracing. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. ACM, New York, NY, USA, 201–212. DOI: http://dx.doi.org/10.1145/3242587.3242633

[32] Tim Oates. 1999. Identifying Distinctive Subsequences in Multivariate Time Series by Clustering. In *Proc. ACM SIGKDD*. 322–326.

[33] Katarina Pantic, Deborah A. Fields, and Lisa Quirke. 2016. Studying Situated Learning in a Constructionist Programming Camp: A Multimethod Microgenetic Analysis of One Girl's Learning Pathway. In *Proceedings of the The 15th International Conference on Interaction Design and Children (IDC '16)*. ACM, New York, NY, USA, 428–439. DOI: http://dx.doi.org/10.1145/2930674.2930725

[34] Michael Polanyi. 1967. The tacit dimension. 1967. *Garden City, NY: Anchor* (1967).

[35] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and others. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[36] Adam Rule, Aurélien Tabard, and James Hollan. 2018. Exploration and Explanation in Computational Notebooks. In *ACM CHI Conference on Human Factors in Computing Systems*.

[37] James Russell. 1980. A Circumplex Model of Affect. *Journal of Personality and Social Psychology* 39 (12 1980), 1161–1178. DOI: http://dx.doi.org/10.1037/h0077714

[38] Rohit Saxena and Niranjan Pedanekar. 2017. I Know What You Coded Last Summer: Mining Candidate Expertise from GitHub Repositories. In *Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17 Companion)*. ACM, New York, NY, USA, 299–302. DOI:http://dx.doi.org/10.1145/3022198.3026354

[39] Richard Sennett. 2008. *The craftsman. [electronic resource]*. New Haven : Yale University Press, 2008.

[40] Kimiaki Shirahama, Lukas Köping, and Marcin Grzegorzek. 2016. Codebook Approach for Sensor-based Human Activity Recognition. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16)*. ACM, New York, NY, USA, 197–200. DOI: http://dx.doi.org/10.1145/2968219.2971416

[41] Philip E. Agre Jeff Shrager. 1990. Routine Evolution as the Microgenetic Basis of Skill Acquistion. (1990).

[42] Robert S. Siegler. 2007. Microgenetic Analyses of Learning. In *Handbook of Child Psychology*. American Cancer Society. DOI: http://dx.doi.org/10.1002/9780470147658.chpsy0211

[43] David Sun, Pablo Paredes, and John Canny. 2014. MouStress: Detecting Stress from Mouse Motion. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 61–70. DOI: http://dx.doi.org/10.1145/2556288.2557243

[44] Katja C. Thoring, Roland M. Mueller, and Petra Badke-Schaub. 2015. Ethnographic Design Research With Wearable Cameras. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '15)*. ACM, New York, NY, USA, 2049–2054. DOI: http://dx.doi.org/10.1145/2702613.2732717

[45] Thierno Tounkara. 2015. Increasing transferability of tacit knowledge with knowledge engineering methods. *Leading Issues in Knowledge Management, Volume Two* 2 (2015), 114.

[46] Sherry Turkle and Seymour Papert. 1990. Epistemological pluralism: Styles and voices within the computer culture. *Signs: Journal of women in culture and society* 16, 1 (1990), 128–157.

[47] Maaike Van Den Haak, Menno De Jong, and Peter Jan Schellens. 2003. Retrospective vs. concurrent think-aloud protocols: testing the usability of an online library catalogue. *Behaviour & information technology* 22, 5 (2003), 339–351.

[48] Gerard Wilkinson, Tom Bartindale, Tom Nappey, Michael Evans, Peter Wright, and Patrick Olivier. 2018. Media of Things: Supporting the Production of Metadata Rich Media Through IoT Sensing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, 206:1–206:13. DOI: http://dx.doi.org/10.1145/3173574.3173780

[49] Nicola Wood, Chris Rust, and Grace Horne. 2009. A Tacit Understanding: The DesignerâĂŹs Role in Capturing and Passing on the Skilled Knowledge of Master Craftsmen. (2009), 14.

[50] Kai Zheng, David A. Hanauer, Nadir Weibel, and Zia Agha. 2015. Computational Ethnography: Automated and Unobtrusive Means for Collecting Data In Situ for Human-Computer Interaction Evaluation Studies. In *Cognitive Informatics for Biomedicine*, Vimla L. Patel, Thomas G. Kannampallil, and David R. Kaufman (Eds.). Springer International Publishing, Cham, 111–140. DOI: http://dx.doi.org/10.1007/978-3-319-17272-9_6