Proceedings of the Workshop on

Designing Technologies to Support Human Problem Solving

DTSHPS'18

Lisbon, Portugal on October 1, 2018, in conjunction with the IEEE Symposium on Visual Languages and Human Centric Computing.

Proceedings Editors: Sandra Fan, Narges Mahyar, and Steven Tanimoto, with thanks to the program committee, listed at the workshop website:

www.cs.washington.edu/dtshps2018

Contents, listed alphabetically by first author:

1. Angela Barriga, Rogardt Heldal and Adrian Rutle (Western Norway University of Applied Sciences, Norway):

A Visual Framework for Transparent and Accessible Machine Learning (Short paper)

2. Alan Blackwell, Luke Church, Ian Hales, Matthew Jones, Richard Jones, Matthew Mahmoudi, Mariana Marasoiu, Sallyanne Meakins, Detlef Nauck, Karl Prince, Ana Semrov, Alexander Simpson, Martin Spott, Alain Vuylsteke and Xiaomeng Wang (Cambridge University, UK and other institutions):

Computer says 'don't know' - interacting visually with incomplete AI models (Long paper)

3. Margaret Burnett, Anita Sarma, Christopher Mendez, Alannah Oleson, Claudia Hilderbrand, Zoe Steine-Hanson and Andrew J. Ko (Oregon State University and University of Washington, USA):

Gender Biases in Software for Problem-Solving (Short paper)

4. Luke Church, Alexander Simpson, Rita Zagoni, Sharath Srinivasan and Alan Blackwell (Cambridge University, UK):

Building socio-technical systems for representing citizens voices in humanitarian interventions (Short paper)

5. Michelle Ichinco (University of Massachusetts Lowell, USA):

Broadening Participation in Online Problem Solving by Increasing Awareness of Common Contributor Qualities (Position paper)

6. Michael Xieyang Liu, Nathan Hahn, Angelina Zhou, Shaun Burley, Emily Deng, Jane Hsieh, Brad A. Myers and Aniket Kittur (Carnegie-Mellon University, USA) :

UNAKITE: Support Developers for Capturing and Persisting Design Rationales When Solving Problems Using Web Resources (Long paper) [Full text not available here, by authors' request].

7. Dastyni Loksa and Andrew Ko (University of Washington, USA):

Problem Solving and the Future of Computing Position Statement (Position paper)

8. Elham Moazzen, Robert Walker, Joerg Denzinger and Lora Oehlberg (University of Calgary, Canada):

Incremental Understanding and Coordination of Software Re-architecting (Long paper)

9. Daniel Rough and Aaron Quigley (University of St. Andrews, Scotland):

Towards End-User Development for Chronic Disease Management (Long paper)

10. Steven Tanimoto and Sandra Fan (University of Washington, USA):

Collaborative Problem-Solving Technologies: A Taxonomy of Issues (Long paper)

A Visual Framework for Transparent and Accessible Machine Learning

Angela Barriga
Western Norway University
of Applied Sciences
Bergen, Norway
abar@hvl.no

Rogardt Heldal
Western Norway University
of Applied Sciences
Bergen, Norway
rohe@hvl.no

Adrian Rutle
Western Norway University
of Applied Sciences
Bergen, Norway
aru@hvl.no

***Abstract*—Machine Learning (ML) is a promising technology, empowering its users with the ability to solve a wide range of problems. Yet, to apply and build a ML system, it is necessary to possess a complex skill set, making it difficult to use and learn by general users. Besides, the black-box nature of ML algorithms make bias prevention a difficult task. Reducing ML learning curve and incrementing its transparency would lead to open this field to society and democratizing its use.**

Our objective is to reduce this complexity and increase bias visibility, hence in this paper we present a visual framework to make ML more accessible and interpretable for users without data science expertise. In this research, we study how Human Computer Interaction, Visual Languages, Auto Machine Learning and Explainable Artificial Intelligence can help in making ML more user-friendly.

I. Introduction

Machine Learning (ML) has risen as one of the most popular technological areas due to its usefulness in a broad range of domains, allowing the automation of problems difficult to solve by humans. Despite its popularity, ML is not yet an accessible field to users without data science knowledge, owing to the complex skill set required to learn and use it, as well as to interpret its results.

The prerequisites for learning ML include knowledge of algebra, calculus, programming and statistics. These subjects are already, as requirements, difficult to learn, which closes the usage of ML to a selected group of users from technological and data science domains (mathematics and statistics).

Some authors have already stated the vital importance of science democratization [1] and helping society with technology's growing skill demand, especially within the ML and artificial intelligence (AI) scope [2]: "AI may play a role in augmenting or reducing the socio-economic impact of intelligence and wealth depending on whether it is sufficiently accessible for a wide population. Tools will be needed to help individuals cope with complexity." Future will demand a society able to function productively in human-technical environments, as ML adoption will affect more and more parts of daily life.

It is important for citizens to be able to understand how ML works in order to be able to identify biased situations [3] and to build critical reasoning regarding this technology (e.g.; ability to identify fake news bots, personalized marketing campaigns, etc.). Democratizing and lowering the accessibility threshold of complex technologies have proven to increase their use and research efforts. Abstracting unnecessary technical details for the user by using Human Computer Interaction (HCI) techniques, building a Visual Language (VL), creating an intuitive User Interface (UI) or following a human centered design can be good approaches to tackle this complexity issue. Some successful examples are visual programming tools for controlling drones [4], the use of Lego blocks in robotics [5] and the well-known block-programming tool Scratch [6]. ML could make use of these approaches to be transformed into an easier to access field.

Another challenge for accessibility is that ML models are many times non-intuitive and difficult to understand [7]. Users get results without knowing why, since algorithms do not provide any reasoning behind their output, leading many times to biased and unexpected situations (e.g; word embedding systems biased by gender stereotypes [8]). Explainable Artificial Intelligence (XAI) aims to break this black-box nature by providing ML results together with explanations about their origin. This could increase ML trustiness in domains with really sensitive data (such as healthcare), empower novice users with easier to understand algorithms and prevent bias by understanding which data features influenced each result.

Much of the complexity for ML users also lies in finding which algorithm provides the best solution for a particular problem and how to tune its hyperparameters. Automated Machine Learning (AutoML) frees users from these burdens [9] by following a ML-based solution able to select the best algorithm according to training data and problem nature, thus reducing the amount of knowledge needed to use ML.

Our goal with this research is to explore the possibilities of ML as an accessible technology in supporting human problem solving. Our contribution is a prototype, that given a training data set is able to select the best algorithm and hyperparameters possible by using AutoML. So, it reduces the burden by a user wanting to use ML. Also, it makes use of XAI to enhance results transparency by providing a detailed explanation of their origin.

At the moment the tool is restricted to classification problems, but our goal is to include other types of ML algorithms. We believe more research is needed in this direction in order

to achieve the true potential of ML, making its processes more visible to tackle bias issues.

II. Related Work

The wish of making ML more user-friendly has led to the appearance of different tools, both commercially and in academia. In this section, we focus on those tools that claim to be easy to use.

Some of them aim to provide accessible ML solutions for domain experts in specific fields. SubCons [10] is an example of a web-based server in where subcellular researchers can perform ML operations thanks to an intuitive UI; another example can be found in [11] where ML can help cyber security experts.

Other projects focus on transforming ML algorithms into VL, making them easier to use. Some examples can be found in Lobe (https://lobe.ai/) and Barista [12] with a strong focus on deep learning networks.

Most efforts aim to build more general tools, suitable for more than one specific domain or algorithm. An increasing number of companies are trying to make ML more accessible, some of them are well-known (Microsoft Azure, Amazon Web Services, Google Cloud, IBM Watson, etc.) and others are small startups, for example H2O (https://www.h2o.ai/) or BigML (https://bigml.com/). For all of them the visual component is key. Some remarkable projects in academia are Orange [13], a drag-and-drop visual ML environment, and PennAI [14], a user-friendly Artificial Intelligence (AI) system making use of genetic programming.

It would be interesting to investigate to which degree these tools are making ML easier for domain experts, although they are good at some of their functionalities (e.g., offering end-to-end solutions or cloud computational power). It is true they lower the amount of ML expertise needed but many of them still require a deep knowledge of the field, since they use specialized technical jargon and black-box algorithms. None of the above mentioned tool offers ML transparency, therefore make bias prevention a difficult task. Also, commercial tools are not suitable for individual researchers nor general users due their cost [14] and also their usage may lead to vendor lock in.

Regarding XAI, there are some projects like Lime [15], an open-source Python library that provides numerical and visual explanations of some ML algorithms results. In [16] authors explore different principles for building understandable XAI systems, with a strong focus on interactive debugging. Other projects like [17] focus on improving context-aware systems intelligibility, providing a solid basis to explore which kind of XAI explanations work better for different scenarios.

III. Research Approach

Our research methodology is design science [18]. Firstly, a gap in the state of the art of ML has been identified (lack of user-friendly visual tools providing accessible ML) by analysing relevant literature and ML tools available online. From it we develop knowledge (how to make ML more accessible) by building and validating an artifact (accessible ML visual framework). The building of said artifact is being tackled by following an incremental development approach, prototyping and iteratively improving its performance. Consequently, the prototype reflects the findings from research as they are found, scaling up iteration by iteration.

In order to evaluate the prototype it is considered necessary to test it with different domains in real scenarios, not only in an academic environment. Connection with real world is of utmost importance in a human-centered research like this. ML competition websites and OpenML (www.openml.org) offer real datasets from trusted sources, ideal for initial testing. More definite evaluation will be done by designing experiments with subjects from different domains, with diverse degrees of data-science expertise. We consider healthcare an especially interesting domain for case studies since the sensitive nature of its data and the usual lack of data science knowledge of its professionals would be perfect to show the possibilities XAI opens.

The evaluation will be measured by comparing the performance of the artifact with traditional ML approaches by using the same data. During the experiments, different features will be taken into account, such as: capacity of the user to work with the visual framework without any extra guidance, adaptability of the tool regarding the problem to solve, overall performance, results interpretability, ability to actually solve the problem proposed, approximation of the time saved using the tool in contrast with traditional ML development and direct feedback, suggestions or feelings from the user. As a result, the general performance of the framework will be analysed, validating whether it has reached its goals or not and improving it if necessary with each development iteration.

IV. Results

At the moment, three techniques have been identified as useful for achieving ML accessibility and transparency: HCI, AutoML and XAI.

HCI helps to design a VL that abstract technical details not needed by users without data science expertise. This language can offer the user a simple way to interact with a complex technology without losing the power behind it. AutoML is able to automatically pick the best algorithm and tune hyperparameters, thus reducing one of the biggest complexity issue in ML systems. XAI is key for providing transparent ML, better results intepretability and a way to ease bias identification.

For XAI implementation we are using Lime, library introduced in Section II. It is still under development and at the moment, it supports explanations for most classifiers and some regression algorithms. The explanations are produced by learning an interpretable model locally around the prediction to explain [15].

We have combined these techniques in the creation of a prototype that allows users to create and test their own ML systems with no previous knowledge needed.

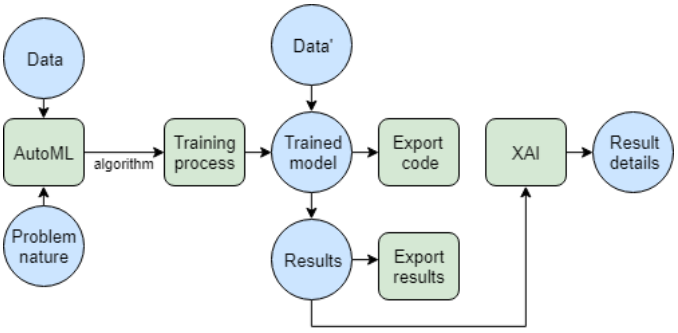

Fig. 1. Workflow of the prototype

Users can learn how to use our tool while using it, since it has been designed with a strong focus on keeping a simple interface with explanations on each screen.

A. Prototype

The prototype consists of a Django web application powered by Scikit-learn, an open source ML library for Python. The UI is created by using CSS and Javascript. The overall workflow of the prototype can be seen in Fig. 1, input/output elements in blue and methods in green. At the moment, it works for classification problems with numerical data and it has the following functionalities:

1) Given a training data set, the prototype selects the best algorithm and hyperparameters possible.
2) Generate and export a trained ML model.
3) Analyse new data, display its results and export them into CSV. Data can be uploaded manually (one row) or using a CSV file.
4) Display explanations for each result generated, indicating which data features led to the prediction and which ones are evidence against it.

The UI has been designed following a human-centered approach abstracting all technical ML language. Users can perform complex ML operations without facing unknown concepts. We have tried as far as possible to make all operations available self-explained (explanations on each screen). UX has been designed so that users can easily discover how the tool works by interacting with it (cursor changes, animations over buttons, etc). There is always only one click needed to switch between screens.

Next images show some screenshots of the prototype. In the images the system makes use of a training dataset for diabetes prevention [19] (publicly available and widely used for research during the last 3 decades). After training, we have uploaded new data for testing and obtained results explanations:

1) Figure 2 displays the training data updated by the user. The upper part of the screen gives information about how that screen works. Here they can select which data will be used and which will be the target of the ML system.

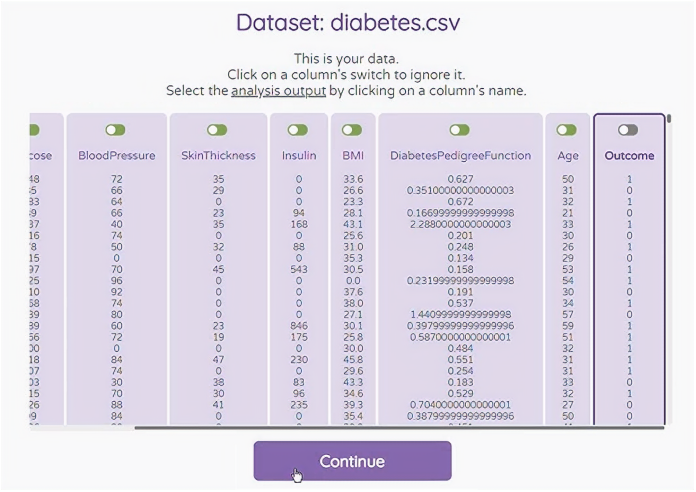

Fig. 2. Screenshot 1: Screen for data input and target selection before training

2) Figure 3 shows results obtained by using the trained model on new data. It displays all rows of the updated data and each result. By clicking on one of them, users can get a detailed explanation of where does that result come from. They can also export all results into a CSV file.
3) Figure 4 displays one result explanation. It provides first the certainty of the result (e.g.; the result reveals that the patient does not have diabetes, with a probability of 62%, meaning the algorithm is not sure about the result and therefore the patient should receive further medical testing). Then it explains which features led to said result (in this example, which factors determine the patient does not suffer diabetes) and which ones showed evidence against it (features that indicate the user could suffer diabetes). This example is especially interesting, since it provides insight about what is exactly causing (or not) diabetes on a patient, which could mean reducing testing time since doctors would know where to focus or providing more specialized and effective care.

V. Conclusion and Future Work

ML is a technology that is increasingly shaping aspects of daily-life, and its influence will increase in the future. It has a strong capability for many problem solving tasks but also to provide biased results. In a desirable scenario, there should be a strong design focus to prevent bias and to make ML more transparent and accessible for the general public.

In order to show how the combination of HCI, VL, AutoML and XAI can help in designing more transparent and accessible ML, we have created a visual framework for user-friendly ML, aimed at users without data science expertise. This software tool has been built guided by the ideas of reducing the level of knowledge needed to work with ML and increasing its results' interpretability.

Our approach is to abstract ML behavior by tackling different levels of its complexity: first by automating ML algorithm

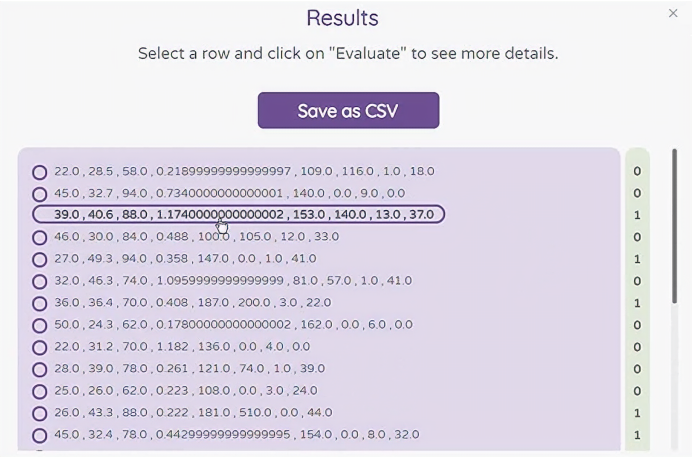

Fig. 3. Screenshot 2: Screen showing analyzed data and results on a trained algorithm

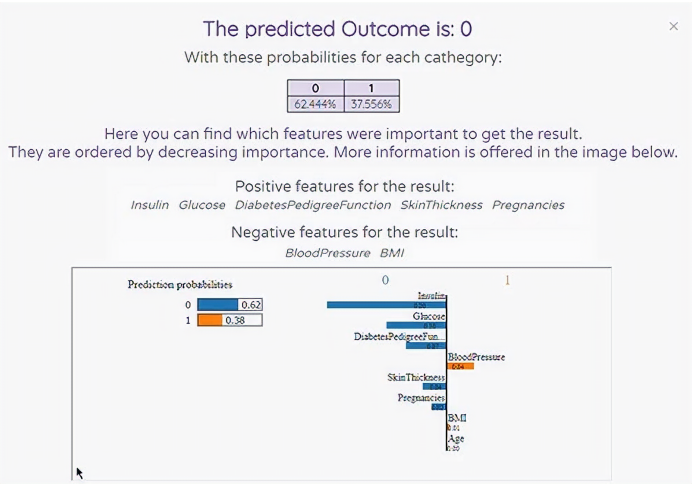

Fig. 4. Screenshot 3: Screen showing the explanation of an specific result

selection and hyperparameter tuning as much as possible, for which we have proposed to use AutoML algorithms. Then, by applying XAI to break the black-box nature of traditional ML algorithms, increase transparency and prevent biased results. Lastly, with a strong focus on the visual part of the framework, by producing a high quality UI and UX that allows users to work with a complex technology in a simple way, without losing any of its possibilities.

Our contribution is to provide users without data science expertise with a tool that allow them to create ML systems on their own, and to fully interpret the results produced by these systems in a easily interpretable format.

As stated in subsection IV-A, the prototype currently works only with classification problems and numerical data, so the nearest future work is to expand the types of data and problems the visual framework can deal with. Then, UI and UX need further improvement and testing, since this is the key part of the framework. Further studying of how users can interact with complex technologies will also be necessary.

Another possible path is to research how ML can be abstracted and personalized for users in specific domains like healthcare, teaching, retail, etc. Finally, it could also be interesting to transform the visual framework into an educational tool in the same way as Scratch.

REFERENCES

[1] M. Brundage, "Economic possibilities for our children: Artificial intelligence and the future of work, education, and leisure." in *AAAI Workshop: AI and Ethics*, 2015.

[2] D. H. Guston, "Forget politicizing science. let's democratize science!" *Issues in Science and Technology*, vol. 21, no. 1, pp. 25–28, 2004.

[3] O. A. Osoba and W. Welser IV, *An intelligence in our image: The risks of bias and errors in artificial intelligence*. Rand Corporation, 2017.

[4] E. Tilley and J. Gray, "Dronely: A visual block programming language for the control of drones," in *Proceedings of the SouthEast Conference*, ser. ACM SE '17. New York, NY, USA: ACM, 2017, pp. 208–211.

[5] P. Herber and V. Klös, "A multi-robot search using lego mindstorms: An embedded software design project," *SIGBED Rev.*, vol. 14, no. 1, pp. 61–70, Jan. 2017.

[6] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 4, p. 16, 2010.

[7] D. Gunning, "Explainable artificial intelligence (XAI)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[8] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," in *Advances in Neural Information Processing Systems*, 2016, pp. 4349–4357.

[9] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 2962–2970.

[10] M. Salvatore, N. Shu, and A. Elofsson, "The subcons webserver: A user friendly web interface for state-of-the-art subcellular localization prediction," *Protein Science*, vol. 27, no. 1, pp. 195–201, 2018.

[11] C. Feng, S. Wu, and N. Liu, "A user-centric machine learning framework for cyber security operations center," in *IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017, pp. 173–175.

[12] S. Klemm, A. Scherzinger, D. Drees, and X. Jiang, "Barista - a graphical tool for designing and training deep neural networks," *CoRR*, vol. abs/1802.04626, 2018.

[13] J. Demšar, B. Zupan, G. Leban, and T. Curk, "Orange: From experimental machine learning to interactive data mining," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2004, pp. 537–539.

[14] R. S. Olson, M. Sipper, W. L. Cava, S. Tartarone, S. Vitale, W. Fu, J. H. Holmes, and J. H. Moore, "A system for accessible artificial intelligence," *CoRR*, vol. abs/1705.00594, 2017.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 1135–1144.

[16] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf, "Principles of explanatory debugging to personalize interactive machine learning," in *Proceedings of the 20th international conference on intelligent user interfaces*. ACM, 2015, pp. 126–137.

[17] B. Y. Lim, "Improving understanding and trust with intelligibility in context-aware applications," 2012.

[18] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[19] J. W. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, "Using the adap learning algorithm to forecast the onset of diabetes mellitus," in *Proceedings of the Annual Symposium on Computer Application in Medical Care*. American Medical Informatics Association, 1988, p. 261.

Computer says ‘don’t know’ - interacting visually with incomplete AI models

Alan Blackwell
Computer Laboratory
University of Cambridge
Cambridge, UK
Alan.Blackwell@cl.cam.ac.uk

Luke Church
Computer Laboratory
University of Cambridge
Cambridge, UK
luke@church.name

Ian Hales
Reach Robotics
Bristol, UK
ian@ian-hales.com

Matthew Jones
Judge Business School
University of Cambridge
Cambridge, UK
mrj10@ cam.ac.uk

Richard Jones
Boeing
Bristol, UK
richard.jones16@boeing.com

Matthew Mahmoudi
Department of Sociology
University of Cambridge
Cambridge, UK
mm2134@cam.ac.uk

Mariana Marasoiu
Computer Laboratory
University of Cambridge
Cambridge, UK
mariana.marasoiu@cl.cam.ac.uk

Sallyanne Meakins
Papworth Hospital
NHS Foundation Trust
Papworth Everard, UK
sallyanne.meakins@nhs.net

Detlef Nauck
Applied Research
BT
Ipswich, UK
detlef.nauck@bt.com

Karl Prince
Judge Business School
University of Cambridge
Cambridge, UK
kjp30@cam.ac.uk

Ana Semrov
Computer Laboratory
University of Cambridge
Cambridge, UK
ana.semrov@gmail.com

Alexander Simpson
Computer Laboratory
University of Cambridge
Cambridge, UK
Alexander.Simpson@cl.cam.ac.uk

Martin Spott
HTW Berlin
Berlin, Germany
Martin.Spott@HTW-Berlin.de

Alain Vuylsteke
Papworth Hospital
NHS Foundation Trust
Papworth Everard, UK
a.vuylsteke@nhs.net

Xiaomeng Wang
Computer Laboratory
University of Cambridge
Cambridge, UK
xw337@cl.cam.ac.uk

***Abstract*—This paper presents a design approach to intelligent user interfaces that purposely undermines the perceived "intelligence" of the automated system, with the intention of improving collaborative problem solving. Our goal is for domain experts to maintain a high level of agency, having confidence in their own judgment wherever appropriate, and easily able to question or supplement actions taken by the automated system. We support this approach with four design case studies that incorporate methods from computer vision, natural language processing, data mining, and exploratory visual analytics. Each of the resulting systems has been designed for a specific context of domain expertise. The design guidance derived from these cases relates to the maintenance of uncertainty through visual design cues, encouragement of judgment decisions by expert users, and emphasising the limited evidential status of partial data sets.**

Keywords—intelligent user interfaces, certainty, explanation

I. Introduction

The comedy sketch show *Little Britain* created the catchphrase/meme: ‘computer says no’. Following a long tradition of satirical responses to bureaucracy, this particular meme economically captures the (literal) mindlessness of the supposedly intelligent computer, the frustration of binary decisions in otherwise nuanced human interaction, and the potential for abdication of human agency in information systems. In collaborative problem solving situations, each of these factors presents a serious obstacle to effective collaboration. In our research, we seek design strategies to mitigate those obstacles. We are particularly concerned with innovative design for new technical developments in artificial intelligence and visual interaction, motivated by theoretical perspectives such as mixed initiative interaction [1] and attention investment [2].

We suggest that a more appropriate design stance for such technologies is *computer says don't know*, to be applied in any situation where information is incomplete or alternative courses of action are available - which is to say, *every* situation involving collaborative problem solving. We offer four design case studies that explicitly address specific everyday issues in intelligent system design: incomplete (training) data, ambiguity or uncertainty in inferred models, and availability of human expertise. Each of the four case studies relates to a technical trend in current AI: i) natural language processing, ii) computer vision, iii) ‘big data’ mining, and iv) exploratory visual analytics. Each has resulted in development of an interactive prototype, intended for use by experts in a particular domain: i) international development aid, ii) forensic policing, iii) business decision making, and iv) clinical medicine.

We briefly describe each of these problem domains, and the design strategies taken to support expert problem solving, in the following sections. There are substantial differences between the four design case studies, with regard to the precision of the statistical models, the completeness of the available data, and the complexity of the interactive visual designs. Nevertheless, in this workshop discussion we draw lessons across the four cases to demonstrate consistent design strategies that rectify the unhelpful ‘computer says no’ attitude. The contributions of this work are as follows:

1. Rather than ‘binary’ category judgements (whether yes/no, or larger numbers of logistic classes), we create visualisations that explicitly maintain ambiguity or uncertainty through graphic design cues.

2. Rather than encourage abdication of human agency, we explicitly require the users to make their own judgement decisions through navigating, labelling or constructing interpretive models.

Mariana is a Vice-Chancellor’s Scholar and is supported by an EPSRC industrial CASE studentship co-sponsored by BT. She is also supported by a Qualcomm European Research Studentship in Technology.

The ICUMAP project is part of the Health Foundation’s Insight programme. The Health Foundation is an independent charity committed to bringing about better health and health care for people in the UK.

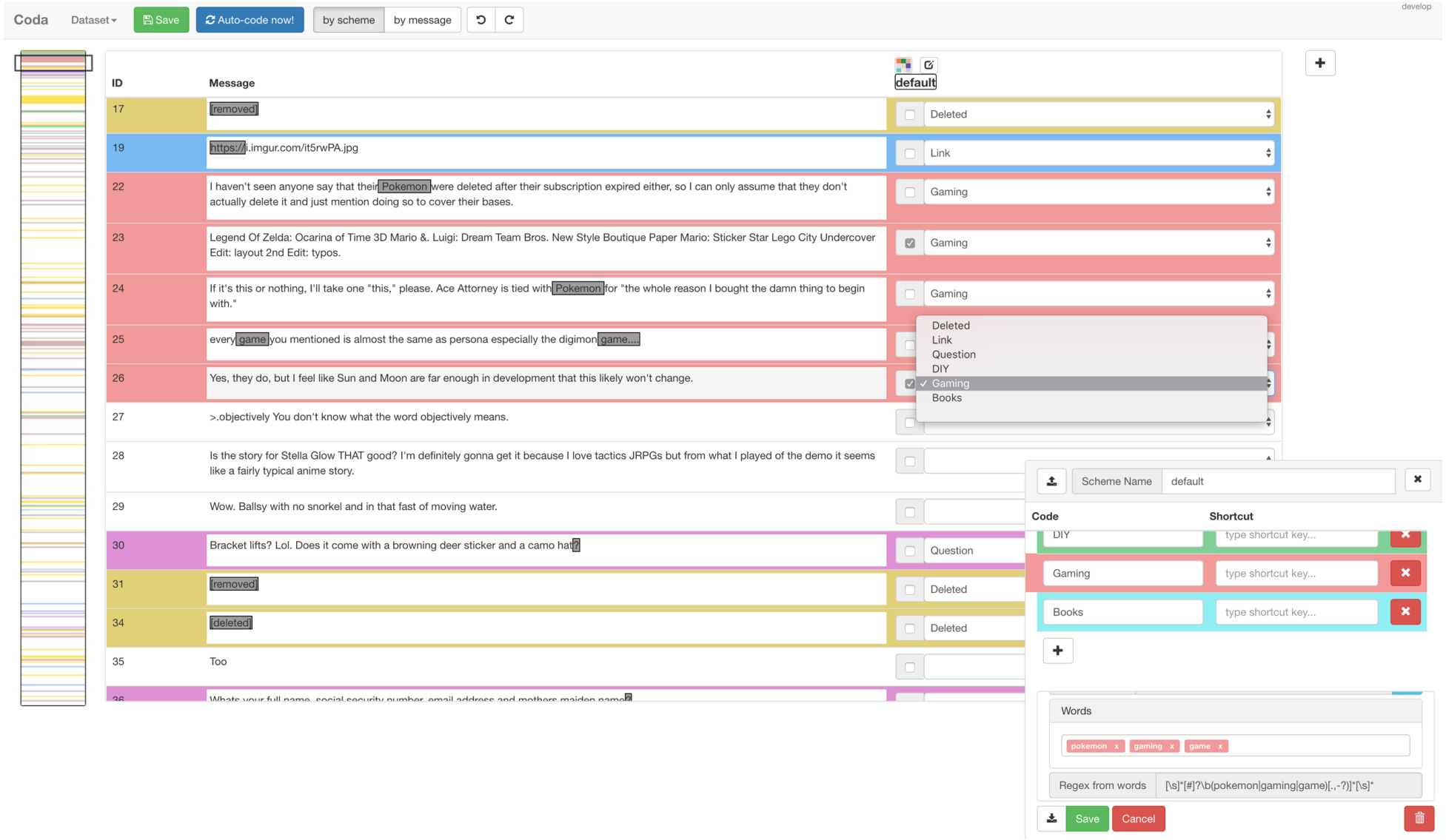

Fig. 1. Coda: each line in the main screen shows a text message, coloured to show how a researcher has categorised it. The available categories (an extensible set) are shown in the inset at lower right. The coloured scrollbar at the left offers an overview of the whole dataset, useful for navigation. The messages are also reorderable to compare categories and confidence in proposed automatic classifications. Since the data Coda is used with is usually sensitive, the data in this screenshot is a sample from the Reddit comment data available on Google BigQuery (https://bigquery.cloud.google.com/table/fh-bigquery:reddit comments)

3. Rather than presenting statistical models as being correct because the data is objective, we draw attention to the ways that the model itself has been created through expectations of usage that frame what can be discussed.

The remainder of the paper discusses each case study in turn, then concludes with a summary of the contribution as demonstrated in those case studies. The studies do represent a wide range of different problem-solving contexts, and we have tried to provide a rich understanding of that context - none of these projects were originally designed to illustrate a simple research theme.

II. Case study 1: Coda

The first case study is a labelling tool designed for an efficient workflow that continually highlights and allows questioning of the categories being constructed, addressing contributions #1 and #2. The application domain relates to the work of Africa's Voices Foundation (AVF), a non-governmental organisation (NGO) whose purpose is to engage with hard-to-reach populations in sub-Saharan Africa through the use of information and communication technologies. AVF works in partnership with local radio stations in remote rural areas, with a focus on conflict regions, low income and low literacy populations. SMS messages from the local population are collected using an SMS gateway and software running on a laptop at the radio station. Community information programmes are accompanied by audience participation surveys, in which listeners are invited to provide a personal perspective on current health issues via SMS. Follow-up questions are sent to listeners who respond to the surveys, requesting demographic information and other survey data. A more in depth description of the methodologies and processes that AVF uses, as well as lessons learnt in designing scalable socio-technical systems for problem solving, is being submitted separately [3].

Analysis of these natural language data-sets is central to the Africa's Voices business model. The "customers" for the analysis are typically humanitarian aid organisations and other NGOs such as United Nations agencies. The business process for Africa's Voices focuses on efficient and reliable coding and analysis of the SMS messages (a difficult process, since the respondents often come from speakers of mixed, low-resource languages), and traceable evidence for communication to the NGO customers. This is the context to our development of an AI-assisted coding tool for use by translators and researchers on the Africa's Voices staff.

The immediate application is a research collaboration between Africa's Voices and UNICEF Somalia, contributing to a programme of drought and famine relief in politically unstable regions of Somalia. In this project, the SMS texts being collected are written in Somali. This introduces further challenges for analysis, as there is no standardised orthography for local Somali speakers, and there are many variant spellings of place-names, as well as diverse cultural attitudes that influence responses to apparently straightforward demographic questions.

Coda is a qualitative coding tool implemented as a Chrome extension (for maximum portability and deployability in diverse environments). The UI supports fast (one-key) decisions that colour-code the data set. As the user works, back-end inference algorithms (currently trivial, but being extended) refine a classifier for semi-automated classification of unseen items. The user can select words in the message to hint to the classifier how they made their decisions.

A colour-coded scrollbar summarises proportion of manually and automated decisions, allowing users to shift between review, correction and refinement to audit and control semi-supervised learning. This colour-coding also serves to explicitly highlight where the computer "doesn't know" how to code a message.

III. Case study 2: ForensicMesh

The second case study proposes an alternative to Computer Vision techniques such as structure-from-motion, by avoiding photorealistic scene rendering in favour of explicitly incomplete models, and highlighting contributions #1 and #3. As more video material becomes available from the extensive usage of Body Worn Cameras (BWCs) in policing, corporate actors have entered the market of building Evidence Management Software (EMS). We anticipate that future EMS systems will use this video to recreate crime scenes and other sites of investigation, as is already done using 3D modelling software and evidential photographs. While there is a plethora of literature and research tackling the adoption of BWCs in policing, work on photogrammetric techniques whereby sites of interest can be reconstructed and verified, and discussions on the implications of Big Data and AI-mediated indicators (see Cheney-Lippold's 'We are data' [4] and Eubanks' 'Automating Inequality' [5]), there appears to be a dearth in research at the intersections of these areas. This has consequences for a range of practices including the emergent practice of predictive policing, counter-terrorism and investigative policing. The ForensicMesh project sought to add value in the process of identifying narratives and storylines as these relate to aggregated video footage from BWCs in policing. With an aim to facilitate a greater space for human judgment in computer-vision aided investigations, as well as for understanding and identifying the distinct and subjective human perspectives of each wearer of BWCs, the project rendered the wearer as a scene element against the backdrop of a static parsimonious scene model, to give analysts access to the human context of data collection.

In high-risk site investigations, BWCs are increasingly used for two possible functions: ongoing monitoring of what are determined to be high-risk sites of potential incidents of terrorism, and captured footage during the investigation of a newly discovered threat or in the aftermath of such a threat. Literature in criminology and sociology has attempted to determine the extent to which the usage of BWCs reduce or increase violence against — and use of force by — the police (see for instance the 2017 report by Barak Ariel [6]). Investigative bodies, such as the NYC Civilian Complaint Review Board, have endorsed the extended usage of BWCs as a step towards greater accountability [7], and there is a general sense that the technology provides objective evidence [8]. According to Wasserman [9], broadly speaking, the position of BWC proponents is commonly summarised into three advantages: 1) 'Video offers unambiguous and objective evidence for all future police-citizen encounters'; 2) 'Video evidence will reduce citizen complains [and] better prove accurate claims and disprove false claims'; 3) 'police and public will behave better knowing that they are being recorded' [9]. While there is plenty to be said for all of these proposed benefits, this project took a point of departure in the first claim.

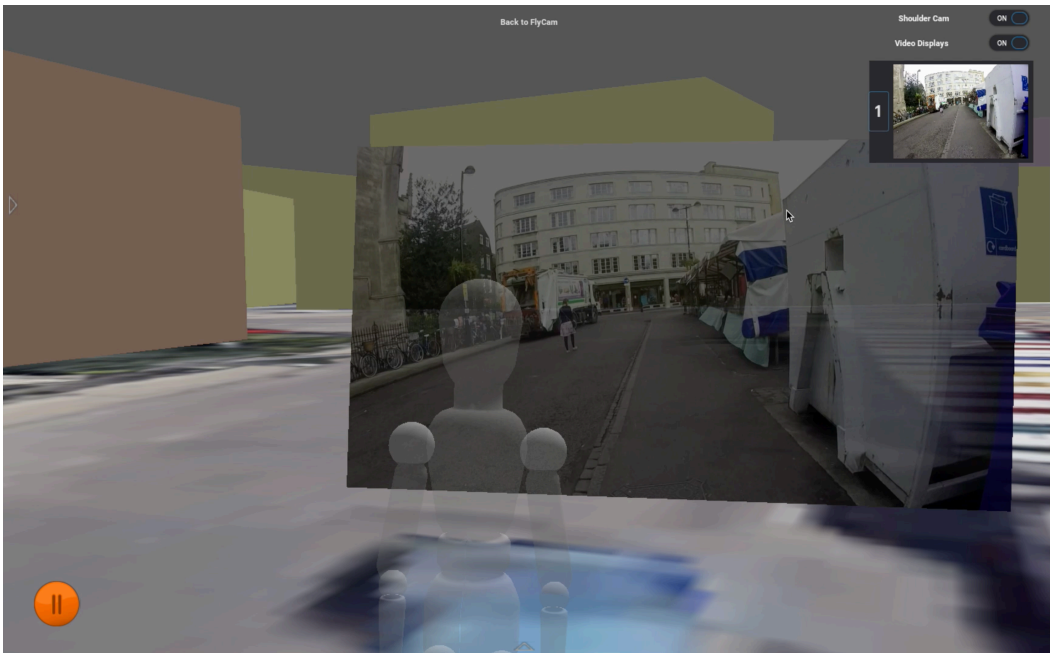

Fig. 2. ForensicMesh: the image shows a parsimoniously rendered citiscape, with only coloured outlines of surrounding buildings. Within the scene, a video plays on a suspended frame, with a human figure in front of it representing the viewpoint of the police officer whose body-worn camera originally captured this video.

Existing research, including by Jones et al. [10], debunk the misconception that BWC footage is "objective", and stress how subjective experiences of incidents shape distinct perspectives of what said footage depicts. Investigative journalists and fact-finders have developed special expertise in creating standards, tools, and methods for using video and image material from official channels as well as publicly available citizen media and other open-source intelligence data. These initiatives (e.g. bellingcat, Forensic Architecture, Amnesty's Digital Verification Corps, Syria Archive etc.) are often attempting to disprove incomplete narratives that may at times be used for political gain (popularly known as misinformation or disinformation). Crucially, they accomplish this by highlighting gaps in the evidence under question, and searching for the "missing link" across a number of different sources. The ForensicMesh project specifically looked at the practices of Forensic Architecture in contemplating environments that would be most revealing of human context and most facilitating of human judgment (and, by extension, doubt).

Forensic Architecture (FA), a visual architecture project group based at Goldsmiths University of London, are widely respected for their contributions to human rights investigations in particular. Their signature immersive 3D reconstructions of sites of violence and crime are particularly worthy of study, as these demonstrate the cutting edge of best practices in establishing visual narratives of events, and filling gaps in data where these exist. Ranging from conflict areas where hospitals or houses may be subject to significant destruction, to murder investigations in Germany, FA use a combination of data-sources including but not limited to: security footage (CCTV), user-generated content (usually in the form of civilian witness footage), images and satellite imagery. Using photogrammetric processing and 3D modelling, FA reconstruct the scene of the particular event in 3D. Videos are hence layered on top of the construction and played in accordance with their sequential timing. Subsequently, linkages between multiple videos and events can be made, and cause and correlation could be established. Virtual sensors such as eye-tracking for subjects in videos can be used to track what event participants are able to see. This approach demonstrates a novel but time-consuming strategy to the reconstructive process. Advancements in computer vision and machine learning, however, mean that the process can be simplified, and — with a critical design and data justice approach — also avoid outsourcing judgment to a machine.

The first aim of this project was to render a 3D reconstruction of the BWC wearer's field of vision, using photogrammetry. This involved the modelling of two primary forms of objects: persistent versus transient objects. Persistent objects include buildings, roads, lights, and signs. Moving, transient objects, on the other hand, include cars, human beings, and animals; these however present a challenge. As transient objects move, their position once outside the field of vision of the BWC wearer can no longer be known and presented with certainty. The predicament presented in such a scenario is evident in the decision between modelling the transient objects in:

1. a predicted position;
2. a fixed position, or;
3. excluding them from the 3D model (once they have left the field of vision). This presents an obstacle regarding the decision to render the scene in 3D or 2D (or both).

In ForensicMesh, we follow FA's approach in the usage of 2D 'video players' embedded within a 3D scene model. A parsimonious 3D scene model of persistent objects (buildings, roads, etc.) can be constructed based on partial information (including images and videos). In our case, we use OpenStreetMap and aerial LIDAR data to generate the parsimonious model. Original footage can then be embedded within a 2D video player to reflect the spatial position and context of the footage, and the wearer of the BWC is represented as a 3D avatar that moves across the scene (Fig 2). The scene can represent multiple wearers with different perspectives of the event. The moving trajectory of each BWC wearer is estimated using a SLAM method [11].

This has several advantages for investigations: First, 2D video players only display what the camera actually captured, leaving no room to display movements and events that were not captured. Second, as the movement of the wearer of the BWC is mapped, a clear timeline of the data-collection event — as well as of the incident itself — emerge, which significantly improves the verification process altogether. Lastly, it emphasizes the temporal and spatial relatedness between multiple BWCs. Where significant gaps are visibly apparent, the analyst is prompted to search for additional sources of evidentiary information beyond the interface.

It is vital to understand the use of tools such as ForensicMesh as a practice that aids — rather than automates — fact-finding processes. 2D video players are deployed where only partial information is captured by BWCs to emphasise new emerging lines of inquiry. In this way, the process of fact-finding becomes a practice of problem solving led by human agents, as gaps prompt the exploration of new leads. These static "gaps" are furthermore an opportunity for the inclusion of open-source intelligence data, including social media, which could shed light on what is not known about the gaps. Explicitly representing these gaps can add value in prompting the investigator to use other resources available to them when the "computer says don't know".

The project set out to develop a photogrammetric tool for video analytics by drawing on best practices in the fields of criminology, forensics, and investigative digital forensics. During the design process, it became apparent that in the development of new and innovative systems for evidence management, it is necessary to build in "uncertainty" by design. Through emphasis on existing or missing connections between BWC footage, ForensicMesh was designed to recenter algorithmically-mediated investigative environments as processes fundamentally of human judgment. This approach not only reiterates the subjective nature of BWC footage, but also demonstrates that ML and computer vision-based technologies can be used outside of regimes that reinforce noxious social biases which are at risk of being algorithmically reproduced.

IV. Case study 3: SelfRaisingData

The third case study is a data visualisation tool for use in the absence of data, highlighting contribution #3. There is an increasing need for business decision making processes to depend on analysing large quantities of data. However, not all the data is easily available or even collected when questions and hypotheses arise, nor is there much time in the fast paced context in which business managers operate to sit down with an analyst and explain and detail the high level question into deliverables. Data analysts have only a short time after they receive an analysis request to clarify the business manager's question. They rely on their expert knowledge of the business domain and of the organisational context to anticipate the (implicit) needs of the business.

The focus of this project was building a data visualisation tool that would support remote collaboration between data analysts and business managers requesting data analyses and reports. Through several interviews with data analysts working for BT, we identified a number of challenges in their existing workflows, from difficulties of data extraction to the importance of careful communication of results to non-experts. A more in depth description of the research methodology and of the tool created can be found in Mărăşoiu et al. [12]; here we describe the part of the analytical process that we chose to design a solution for, and briefly describe the technical artefact created, emphasizing the three design strategies discussed in Section 1.

In this project, we chose to focus on the hypothesis clarification and refinement part of the analytical process - the conversation between the analyst and the requester, where the former aims to better understand the question being asked by the latter.

Some of the analysts we interviewed work on many small data analysis projects, often at the same time. High level business questions such as "why is [this region] not as good as everybody else, what is happening there" are the typical starting point for such a project. But before the analysts can get started on finding and extracting the needed data from the company's databases and data silos, they need to both have a better understanding of what the requester needs, and to turn the high level question into actionable steps and outcomes. They need to add parameters to the original hypothesis, and "fill in all the details", by asking more questions about e.g. which aspects of the region they should look into, what "everybody else" means, what "not as good" means. The data analysts we interviewed had these kinds of conversations primarily through phone calls and sometimes via emails.

Our system, SelfRaisingData, allows sketching and modifying data visualisations in order to support these remote conversations structured by analytic hypotheses.

Since the analysts we interviewed had such conversations primarily through phone calls, the typical scenario would be that both the analyst and the manager would be working on the same visualisation document on their own computers, whilst on the phone with each other. The visualisation has the role of being an external representation of the analyst's understanding of the manager's question, with the manager being able to comment and point out where their understanding of the question diverges. As such, the analyst can create and edit the visualisation to reflect their understanding, whereas the manger can only annotate it.

Fig. 3 illustrates the system. The choice of visualisation comes from the domain in which the analysts we interviewed worked in, as a large part of the type of data they work with is timeseries data. The central area of the system represents a timeseries visualisation of synthetic data. The data is generated from additively composing a set of parameterisable functions added from a tool panel to obtain a trend line. Each component function can be parameterised independently in a function editor by dragging the value handles of its properties on axes corresponding directly to the axes of the final visualisation. The user can add, remove, and modify each component function.

Whilst the resulting composed function could be displayed as a line chart, this visualisation style can result in users fixating on manipulating the parameters of the composing functions in order to achieve a smooth line. Instead, we add noise to the trend line by 1) transforming the continuous function into a discrete set of points by sampling N equally-spaced time coordinates (the X axis) and 2) sampling the value coordinates (the Y axis) from standard distributions having the value of the trend line function as their mean and a constant fixed variance. To further suggest sketchiness and imprecision, we represent each individual point as a hand drawn cross.

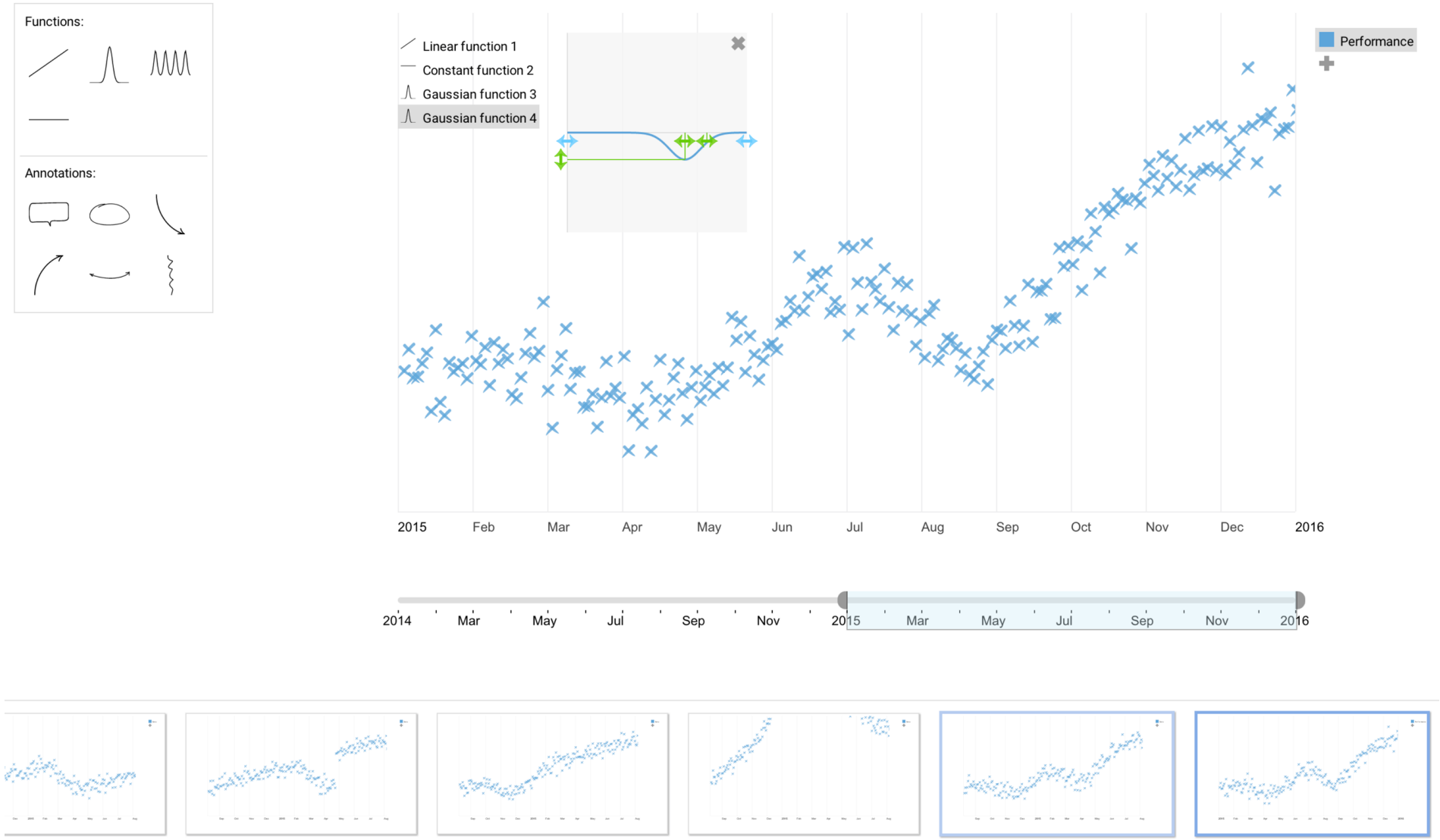

Fig. 3. SelfRaisingData: the time series chart contains synthetic data generated around a trendline through function composition of basic functions (linear, constant, squared exponential and periodic). Each component function can be independently modified in an interactive function editor. The history panel at the bottom contains snapshots of the evolution of the visualisation, allowing the analyst to revert to a previous version and explore alternative hypotheses.

A panel at the bottom of the screen records the history of the visualisation in conceptual sequences of steps (e.g. adding, removing, and editing a different component function, adding a new timeseries). The users can change their mind at any point, having the ability to load any of the past versions of the visualisation and create a new development branch by editing the older version.

Our design goal in this project was to support a conversation about data in the absence of real data through sketching. In this case study, the computer "doesn't know" anything, but provides support for collaboration and statistical problem specification. The visualisation system is a conversational aid that frames the conversation between two people. For example, treating the visualisation as a composition of independent elements (e.g. trends, periodicity, plateaus, dips and peaks) is a deliberate design choice. It adds hypothesis semantics to the visualisation as each component function acquires an individual meaning (e.g. the March-June performance plateau, the 8th of October sales drop). Further, independently manipulated component functions are still available for discussion even after being composed with other functions. The always-visible list of component functions also draws attention to the way that the sketch has been constructed.

Since the data visualised is synthetic and created by the user, emphasizing ambiguity is also relevant. We achieve this through the noisiness of the scatter plot, which also allows for imprecision when adjusting the parameters of the component functions. This means that sketching can be done quicker, as (spending time to achieve) precision is actively discouraged. Vagueness is further encouraged by representing each point as a hand drawn cross and removing any numbering from the function editor panel. The visualisation is a rough sketch of how the real data might look like.

V. Case study 4: ICUMAP

Our final case study is an interactive visualisation that was created to support clinical judgments in an intensive care unit (ICU) through reuse of electronic health record (EHR) data, highlighting contributions #2 and #3. During treatment in an ICU, large amounts of data are collected for each patient, including both nursing observations and automated data acquisition from monitoring instruments (e.g. blood pressure and pulse) at the bedside. Subsets of this data are collated from hospitals around the UK by the national intensive care registry (ICNARC), which uses it to calculate statistical measures of patient condition for comparison to treatment outcome. However, our research with clinicians across multiple hospitals suggested that these measures have low predictive power and are never used directly to guide treatment, perform triage, alert potential emergencies, or otherwise guide clinical judgment.

We had access to 10 years of data, covering the treatment of 20,000 patients, from an ICU specialising in cardiothoracic surgery (e.g. arterial bypass grafts, heart valve replacements and heart transplants). While standardised statistics such as ICNARC are compiled based on a small number of physiological measures at admission time, we were able to pay attention to how the patient's condition changes during the time they are in the unit. In particular, our clinical collaborators wanted to know when a patient's condition changes in a way that is likely to have adverse outcomes - expressed to the design team as a 'traffic light' indication.

The system we designed, ICUMAP, is a dimension-reduced visualization (Fig. 4), in which a variant of t-SNE [14] is used to construct a reference 'map' of regions in which intensive care patients are 'similar' within a multi-dimensional space of variables monitored during their treatment. The condition of each patient is mapped to a new location at 6-hour intervals, and these points are joined to form a trajectory. Early experiments confirmed that some regions in the t-SNE cluster map were associated with high mortality, meaning that the 'traffic light' goal could apparently be expressed as places where a patient appeared to be 'moving toward' a high mortality region in t-SNE space. However, this is not a mathematically well-formed question. t-SNE can be interpreted as a projection of a multi-dimensional space, but the projection is not necessarily monotonic in any dimension. Our experiments confirmed warnings of Wattenberg et al [15], that possible "tendencies" were simply a tangle of overlaid random lines.

We therefore modified the t-SNE algorithm, adding two new constraints to the distance function. The first was to penalise long distances between successive measurements for the same patient, creating temporal locality as a basis for thinking about trajectory. The second was to promote proximity for measurements taken at the point where a patient had died, meaning that these were more likely to cluster together, with the result that mortality would correspond to particular "places" within the optimised layout. The third was to include the surgery that the patient had undergone as a strongly weighted factor, meaning that cases tended to be grouped according to procedure, corresponding to natural classifications used by clinicians.

We optimised the visual rendering to convey local detail of individual trajectories, while also offering a distribution overview of thousands of these. This involved manipulation of hue gradients, line widths, and alpha (transparency) values so that each trajectory could be viewed as a progression from condition at admission (blue at the start of the line) to either mortality or discharge (red or green). Within the clusters of different surgery types, those types that are more risky can be identified by greater density of red trajectories.

This design reflects a Bayesian approach to clinical decision making. Rather than claiming statistical likelihood of treatment outcomes, we focused on assisting clinicians by improving ease of access to relevant prior cases among the thousands they might consider, and making this information more readily available as a counter to the usual heuristic biases in clinical judgment. We therefore focused on selecting a small number of comparator cases, sufficiently similar to suggest relevance to a patient currently under consideration, but presented within an interpretive metaphor that would facilitate reflection on the current case, while not over-determining the conclusions that might be drawn.

ICUMAP includes many features for interaction with the data archive, always aiming both to provide users with statistical overviews while comparing and contrast a patient currently being treated with cases from the historical records. When mousing over the map, the trajectories for individual patients are highlighted. The measured values for that patient at this time are shown on histograms showing overall distributions, so users can see at a glance how typical this patient is. When one histogram is selected, a mask over the map visually fades areas where the value of this variable is low.

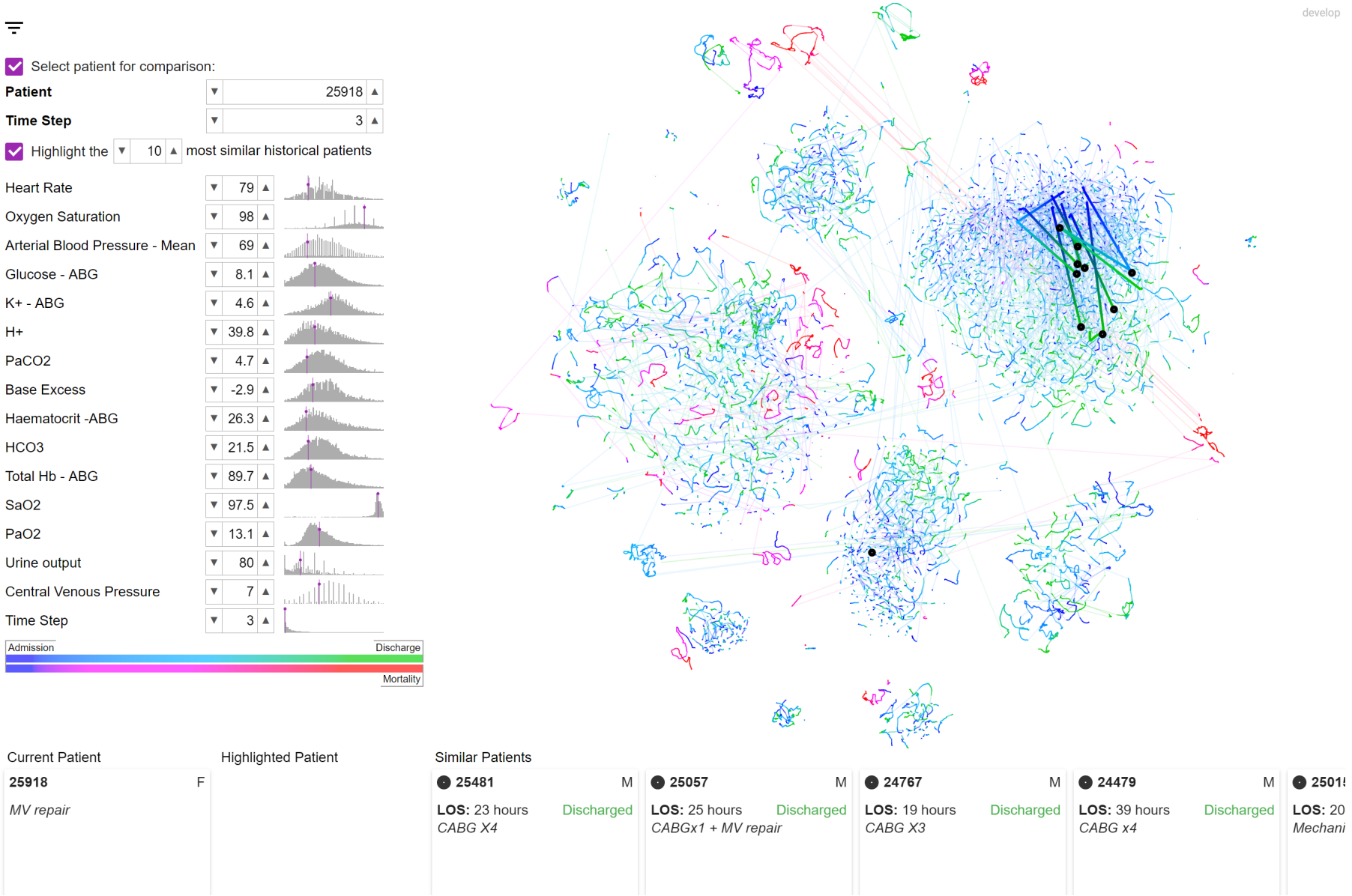

Fig. 4. ICUMAP: the "cloud" shows trajectories of thousands of individual patients within a dimension-reduced multivariate space. Clusters correspond to surgical procedures. At the left, histograms show distribution of values for each of the principal variables, across the whole dataset. At the bottom, patient records are shown for 10 patients whose condition was most similar to the one currently selected by the user.

These features allow clinicians to review prior likelihood, variability, value exceptions, and other broad statistical properties of the large database. However, it is necessary to keep in mind that the t-SNE 'landscape' does not necessarily provide any continuity with respect to a single variable – the variation in masks and intensity is therefore rendered with large tiles to reveal this blotchy variability, and discourage misleading interpretation.

In this final case study, we explicitly evaluated the extent to which our design approach is understood and appreciated by domain experts. This involved a focus group with expert staff, and controlled task observation with students.

A. *Focus Group Evaluation*

We invited eight clinicians and a hospital information analyst to a focus group workshop, at which we demonstrated the functionality of ICUMAP, explaining each aspect of system functionality before pausing for comments and discussion.

We report two primary topics of discussion that arose. The first was the question of which data in the EHR database has the most clinical value for predicting or modifying treatment outcomes. There were diverse opinions, with some senior clinicians strongly advocating use of particular measurements (some of which are not presently recorded in the EHR at all). The second topic focused on the main theme of this paper, which was skepticism regarding the 'sales' message of predictive analytics. Many participants had encountered products that claimed to deliver predictive functionality through multivariate data mining. Clinicians were skeptical that such prediction was possible. Their view was that single variables reflect critical aspects of patient condition, and that multivariate analysis (and hence dimension reduction visualisations) does not significantly add to clinical judgment.

Nevertheless, our central design strategy, drawing clinical attention to a small number of previous cases similar to the current patient, appeared to be welcomed. Scepticism about the value of predictive analytics was directed at other systems (or speculation about what our system might be), while there was productive conversation about the identified similar patients.

B. *Controlled experiment evaluation*

We recruited six participants to evaluate ICUMAP in a controlled task. Three were clinical professionals (two medical students nearing the end of their studies, and one registrar intensivist), and three students from non-medical (engineering, physics, computer science) backgrounds. A predefined data set was loaded, and each participant worked through the same series of interpretive tasks. At the start of each task, the participant was asked an interpretive question without prompting them about ICUMAP functionality. If they had not recognized the expected functionality, the relevant system function would be explained (using a predefined text) before proceeding.

In order to compare interpretation of the t-SNE cluster visualization to more conventional statistical visualisations, participants were first shown a screen with only the histogram distributions. The participant was asked questions regarding their interpretation of these historical distributions, and then shown values for a small number of individual test patients drawn from the database to represent distinctive types, before being offered the opportunity to compare these individual patients to the overall population. The clinical participants were asked how they would interpret the condition of each test patient in their clinical judgment. After making their interpretations, the clinical participants were additionally asked to report how confident they were.

The t-SNE cloud visualization was then revealed, with the explanation that this represented change over time for the same measurements. Participants were asked for their unprompted interpretation of design elements. If they did not volunteer key aspects (time-courses, proximity, mortality), these were explained. Features were tested in turn, each time offering an opportunity for the participant to make their own interpretation before the design was explained. Finally, a small number of patients were selected, each chosen to represent a particular type of surgery or outcome. Participants were asked for their interpretation of likely treatment outcomes, taking into account other patients automatically highlighted as 'similar'. At each point where participants offered an interpretation, they were asked to quantify their level of confidence in that judgment.

We found that while all technical participants recognised histograms as describing statistical distribution, two medical participants initially *misinterpreted* histograms as representing change of a measure over time (the existing EHR system presents a patient overview with prominent time-series graphs). Once they understood the principle, they were able to use the data for assessments of a single patient. However, they relied on *prior expectation* of typical values (i.e. a value range learned during their studies) for initial assessment. Where they had less prior knowledge, they paid more attention to the plotted position of a value within the overall distribution. They used a time step control to explore progression and discuss changes in the patient's condition over time, for example a crisis at one time step. They expressed more confidence in judgments when exploring this historical data.

Technical participants immediately recognized that the ICUMAP visualization was a dimension-reduced view of multivariate data. None of the medical participants recognized this, and found the visual complexity overwhelming. After using the mouse to explore trajectories, they were able to identify properties of the visualization, although one remained uncomfortable throughout the session. All understood that the lines represented the trajectory for a patient, and that red and green reflected mortality. None recognized the basic principle of similar points being near each other. Two of the three recognised without prompting that clusters reflected type of surgery.

Medical participants, in interpreting the overall structure of the cloud, tended to make comparisons between clusters. One expected (incorrectly) that larger clusters might reflect wider distribution of values, while others observed correctly that cluster size related to the number of patients in that cluster.

The key principle of selecting and plotting a group of similar patients was recognized without prompting by all medical participants. When asked to make judgments based on this visualization, they did, as intended, immediately start to make comparisons, for example by starting to talk about relative length of stay, that they did not do when considering histograms alone. A major concern of ours was to avoid over-interpretation of the similarity as predictive data. None of the participants expressed a confidence of 100%, with most

judgments being in the 50% to 80% reflecting a suitable degree of caution. One of the test patients, having healthy values and routine surgery, where all similar cases had been discharged successfully, led a participant to give a 90% assessment that this patient would also be discharged. These findings are encouraging, however, we noted a trend that the confidence judgments tended to increase over the course of the experiment, suggesting that growing familiarity with a tool may still lead to errors of the kind that we wish to avoid through our design philosophy.

VI. Discussion

In this section, we summarise the design strategies that have been applied in these systems, relating them to the three broad contributions outlined in the introduction to the paper.

Firstly, we suggest using graphic design cues to maintain ambiguity or uncertainty in presenting inferred information to users. This is a corrective to the increasing tendency in many machine learning systems to present the categorical output variables of logistic regression as simplistic either/or alternatives, replicating the errors that were systematically identified in the seminal work by Bowker and Star [16]. In Coda, we use desaturated colours to contrast automated suggestions with human-assigned labels, allowing natural interpretations such as complete desaturation (white) being equivalent to no judgment at all, while near-full saturation indicates that the system has identified duplicates that are safely amenable to trivial automation. In ForensicMesh, we eschew photo-realistic scene rendering in order to remind viewers that a geometric model is based only on a persistent coordinate system, not fully-observed state.

Secondly, we suggest that users should be explicitly required to make their own judgement decisions. In a labelling system such as Coda this is trivially true. However, we should recall that most AI systems intentionally hide the labelling phase (usually done offline, via a different interface, and prior to system operation). In ICUMAP, we do not directly present statistical regression on patient condition as a basis for decision making, instead emphasising the clinician's responsibility to retrieve and consider other cases - based on the particularity of clinical interventions and patient case histories.

Thirdly, we suggest that systems draw attention to the ways that the model itself has been created through human processes, reminding users that these processes anticipate the ways the model can be used. The most extreme is SelfRaisingData, in which users are invited to completely "fabricate" data to reflect their ideas about the model. In ForensicMesh, we insert an "observer" into the scene, to emphasise that BWC video is not objective, but reflects the viewpoint of the person who was wearing the camera. In ICUMAP, the use of a query / recommendation interaction paradigm means that the "model" is transient, presented only as a byproduct of the user's brushing over a cloud of patient journeys, or over distributions of measurement values.

Each of these design strategies has potential for use in other collaborative problem-solving settings, and we look forward to workshop discussion considering analogies to other intelligent interaction scenarios.

VII. Conclusion

Although "computer says no" was introduced as a comedy trope, the extension of algorithmic decision making throughout society has become tragic, as when a British man was denied an ambulance because the triage algorithm determined that his case was not serious, despite the fact that he was in agony, correctly diagnosed his own condition, and subsequently died. Cheney-Lippold [4] quotes the operator "We cannot override this, and although there are paramedics in the control room for us to ask, I would not think the system would come up with the wrong answer"

Our four case studies all involve use of data in mission-critical or safety-critical settings. Enhancing reliability of data analysis in such settings is obviously an important research goal for data science and AI. However, at present, these are domains where expert human judgment is respected and human experts take responsibility (and liability) for their interpretations and decisions. This paper has considered a number of visual language design approaches through which expert responsibility can be maintained, with 'intelligent' analysis focused on making the necessary data salient and easily available for human judgment, rather than taking automated decisions.

References

[1] E. Horvitz, "Principles of mixed-initiative user interfaces," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 1999, pp. 159–166.

[2] A. F. Blackwell, "First steps in programming: A rationale for attention investment models," *Proceedings - IEEE 2002 Symposia on Human Centric Computing Languages and Environments, HCC 2002*, pp. 2–10, 2002.

[3] L. Church, R. Zágoni, A. Simpson, S. Srinivasan, and A. Blackwell, "Building socio-technical systems for representing citizens voices in humanitarian interventions," submitted to *Designing Technologies to Support Human Problem Solving - A Workshop in Conjunction with VL/HCC 2018*, Lisbon, Portugal.

[4] J. Cheney-Lippold, *We Are Data: Algorithms and The Making of Our Digital Selves*. NYU Press, 2017.

[5] V. Eubanks, *Automating Inequality: How High-Tech Tools Profile, Police, and Punish the Poor*. St. Martin's Press, 2018.

[6] B. Ariel, A. Sutherland, D. Henstock, J. Young, and G. Sosinski, "The Deterrence Spectrum: Explaining Why Police Body-Worn Cameras 'Work' or 'Backfire' in Aggressive Police–Public Encounters," *Policing: A Journal of Policy and Practice*, vol. 12, no. 1, pp. 6-26, 2017.

[7] Civilian Complaint Review Board, "Civilian Complaint Review Board Issues 2017 Annual Report," New York City, 2018.

[8] D. K. Bakardjiev, "Officer Body-Worn Cameras - Capturing Objective Evidence with Quality Technology and Focused Policies," *Jurimetrics*, vol. 56, no. 1, pp. 79–112, 2015.

[9] H. M. Wasserman, "Recording of and by Police: The Good, the Bad, and the Ugly," *J. Gender Race & Just.*, vol. 20, p. 543, 2017.

[10] K. A. Jones, W. E. Crozier, and D. Strange, "Believing is Seeing: Biased Viewing of Body-Worn Camera Footage," *J. Appl. Res. Mem. Cogn.*, vol. 6, no. 4, pp. 460–474, Dec. 2017.

[11] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, 2018.

[12] M. Mărășoiu, A. Blackwell, A. Sarkar, and M. Spott, "Clarifying hypotheses by sketching data," in *EuroVis 2016 - Short Papers*, 2016.

[13] N. Boukhelifa, A. Bezerianos, T. Isenberg, and J. D. Fekete, "Evaluating Sketchiness as a Visual Variable for the Depiction of Qualitative Uncertainty," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 12, pp. 2769–2778, 2012.

[14] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[15] M. Wattenberg, F. Viégas, and I. Johnson, "How to use t-SNE effectively," *Distill*, vol. 1, no. 10, p. e2, 2016.

[16] G. C. Bowker and S. L. Star, "Sorting Things Out: Classification and Its Consequences". MIT Press, 2000.

Gender Biases in Software for Problem-Solving

Margaret Burnett[1], Anita Sarma[1], Christopher Mendez[1],
Alannah Oleson[1,2], Claudia Hilderbrand[1], Zoe Steine-Hanson[1], Andrew J. Ko[2]

[1]Oregon State University
Corvallis, Oregon

[2]University of Washington
Seattle, Washington

{burnett,anita.sarma,mendezc,olesona,minic,steinehz}@oregonstate.edu, ajko@uw.edu

***Abstract*—The workshop call raises the question of how we can help users problem-solve, especially when the problem to be solved is complex. One answer to this question is to change the way we go about building such systems. Why: most software has extensive biases against certain cognitive problem-solving styles—especially those styles preferred by more women than men. In this position paper, we consider the workshop call's discussion questions from the perspective of GenderMag, a method to pinpoint gender biases in user-facing software that aims to help people problem-solve.**

Keywords—GenderMag, gender biases, problem-solving

I. Introduction

In raising the issue of how we can help users' problem solve, the workshop call suggested a number of questions for discussion. Among them were the following:

- Diversity: What provisions should be made to promote diverse community engagement in problem solving, designing for inclusion across identities as they may relate to aspects such as socio-economic status, gender, culture, etc.?
- Workflows: What social structures and workflows may need to be supported?
- Education/Training: How should students and professionals be educated and trained in order to be able to function most productively in these new human-technical environments for problem solving?

In this position paper, we consider these questions from the perspective of gender biases that affect problem-solving.

II. GenderMag: Diversity and Workflows

We have been working on a method that aims at the first two questions above, Diversity and Workflows. The GenderMag method (Gender Inclusiveness Magnifier) [8] captures individuals' diversity in cognitive styles, especially those styles that tend to cluster by gender. Using this method, software teams can pinpoint gender biases relating to problem-solving styles in the software they are building that tries to support people's problem-solving activities.

GenderMag's foundations lie in research on how people's individual problem-solving strategies sometimes cluster by gender. GenderMag focuses on five facets of problem-solving:

(1) Motivations: More women than men are motivated to use technology for what it helps them accomplish, whereas more men than women are motivated by their interest in technology itself [1, 4, 6, 11, 18, 21, 23, 24, 36]. (2) *Information processing styles*: Problem-solving with software often requires information gathering, and more women than men gather information comprehensively—gathering fairly complete information before proceeding—but more men than women use selective styles—following the first promising information, then backtracking if needed [9, 14, 28, 29, 33]. (3) *Computer self-efficacy:* Women often have lower computer self-efficacy (confidence) than their male peers, and this can affect their behavior with technology [1, 2, 3, 4, 6, 17, 19, 22, 24, 30, 32, 37]. (4) *Risk aversion:* Women tend statistically to be more risk-averse than men [13, 16, 39], and risk aversion can impact users' decisions as to which feature sets to use. (5) *Styles of Learning Technology:* Women are statistically more likely to prefer learning software features in process-oriented ways, and less likely than men to prefer learning new software features by playfully experimenting ("tinkering") [3, 4, 10, 12, 21, 34]. Any of these differences in cognitive styles is at a disadvantage when not supported by the software.

GenderMag brings these facets to life with a set of four faceted personas—"Abby", "Pat(ricia)", "Pat(rick)" and "Tim" (Fig. 1). Each persona's mission is to represent a subset of a system's target users as they relate to these five facets.

GenderMag intertwines these personas with a specialized Cognitive Walkthrough (CW) [38, 40]. The CW is a long-standing inspection method for identifying usability issues for new users to a program or feature. In a GenderMag CW, evaluators answer a question about each subgoal one of the personas might have in a detailed use-case, and two CW questions about each action, using the persona's five facets.

The questions are:

SubgoalQ: Will <persona> have formed this subgoal as a step to their overall goal? (Yes/no/maybe, why)

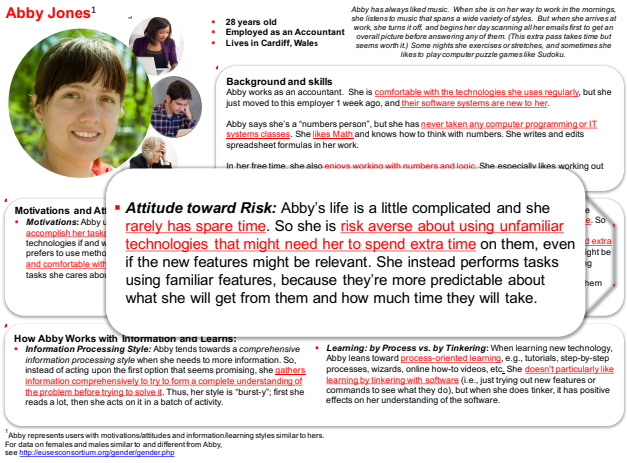

Fig. 1. Abby is a "multi-persona", meaning that she has multiple appearances and demographic portions of her are customizable [31]. One of the facets is blown up for legibility.

ActionQ1: Will <persona> know what to do at this step? (Yes/no/maybe, why)
Action Q2: If <persona> does the right thing, will s/he know s/he did the right thing & is making progress toward their goal? (Yes/no/maybe, why)

Evaluations of GenderMag's validity and effectiveness have produced strong results [5, 7, 8, 15, 20, 25, 26, 35].

III. GenderMag in Education, Training, Integrating into the "Day Job"

One issue with current diversity and inclusion approaches is that they tend to be isolated from other activities in education environments and workplaces. For example, there are special committees on diversity and inclusion, special training sessions on diversity and inclusion, special CS classes on ethics/social issues of computing, and so on. One of our goals is to find ways to *integrate* support for diverse problem-solving into students' everyday education and work tasks.

To help mainstream support for diversity and inclusion into software through CS education, we are working on adding notions of supporting diverse cognitive styles into mainstream CS classes in higher education. We posit that integrating education on how to design for diverse cognitive styles into classes that teach design can help to show that supporting diverse problem-solving styles is *part of* software design, not something "extra".

Toward this end, in a collaboration between Oregon State University and University of Washington, and with the help of nine teacher-researchers across the U.S., we embarked upon an investigation of how to teach the GenderMag method in ways that integrate gender-inclusive software design into CS courses [31]. Analysis of the teachers' observations and experiences, the materials they used, direct observations of students' behaviors, and multiple data on the students' own reflections on their learning revealed 11 components of pedagogical knowledge that affect teaching GenderMag in CS classes. These include strategies for anticipating and addressing resistance to the topic of inclusion, strategies for modeling and scaffolding perspective taking, and strategies for tailoring instruction to students' prior beliefs and biases.

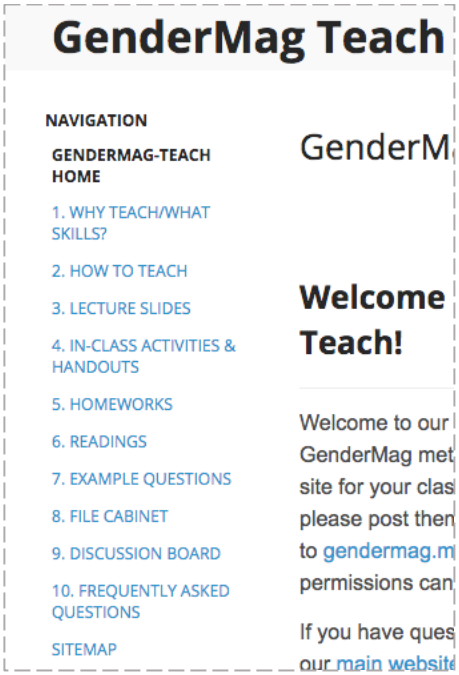

Fig. 2: Structure of the GenderMag-Teach community wiki. Available in full at the GenderMag site (http://gendermag.org/).

The GenderMag-Teach effort is a community, and we invite all interested educators to join us at its community wiki (Fig. 2), which contains downloadable, educator-contributed materials to support educators' efforts in this direction. (http://gendermag.org, then click on the Teaching link.)

IV. GenderMag's Open Source Tool

For people not in education environments— or for those who simply prefer experiential learning over classroom learning— we have developed a GenderMag Recorder's Assistant tool [27]. The tool not only semi-automates software professionals' use of GenderMag, it also walks them through evaluations of the user-facing software they are creating, step-by-step, hands-on.

To use the Recorder's Assistant, a software team navigates via the browser to the app or mockup they want to evaluate, then starts the tool from the browser menu. The main sequence is to view a persona (Fig. 3(c)) and proceed through the scenario of their choice from the persona's perspective, one action at a time. At each step, the tool's "context-specific capture" captures screenshots about the action the team selects (Fig. 3(a)), and records the answers to questions about it (Fig. 3(b)). The tool saves this sequence of screenshots and questions/answers to form a gender-bias "bug report."

The Recorder's Assistant is freely available on Open Source,

Fig. 3: The Recorder's Assistant tool during an evaluation of a mobile time-and-scheduling app. (Left): The app being evaluated is displayed with (a) a rectangle around the action the evaluators are deciding if a user like "Abby" will take. (Right): A blow-up of portions of the GenderMag features for the app: (b) the GenderMag question the team is answering at the moment, including a checklist of Abby's facets; and (c) a summary of the persona the team has decided to use (in this case, Abby).

and anyone can download it and/or contribute to it (http://gendermag.org, click on the Tool link).

V. Calls to Action

This position paper is a call to action: in researching ways to help people problem-solve, we must all keep in mind diversity of cognitive styles. Cognitive diversity is what makes teams, businesses, and society the most effective they can be in solving problems. Also, in more concrete calls to action, we invite you to http://gendermag.org to join our collaborations to create better software and better education for everyone.

Acknowledgments

This work has been supported in part by NSF grants 1528061, 1559657, 1560526, 1703304, and 1735123.

References

[1] L. Beckwith and M. Burnett, Gender: An important factor in end-user programming environments? IEEE VL/HCC, pp. 107-114, 2004.

[2] L. Beckwith, M. Burnett, S. Wiedenbeck, C. Cook, S. Sorte, and M. Hastings, Effectiveness of end-user debugging software features: Are there gender issues? ACM CHI, pp. 869-878, 2005.

[3] L. Beckwith, C. Kissinger, M. Burnett, S. Wiedenbeck, J. Lawrance, A. Blackwell, and C. Cook, Tinkering and gender in end-user programmers' debugging, ACM CHI, pp. 231-240, 2006.

[4] M. Burnett, L. Beckwith, S. Wiedenbeck, S. D. Fleming, J. Cao, T. H. Park, V. Grigoreanu, and K. Rector, Gender pluralism in problem-solving software, Interacting with Computers 23(5), pp. 450–460, 2011.

[5] M. Burnett, R. Counts, R. Lawrence, H. Hanson, Gender HCI and Microsoft: Highlights from a longitudinal study, IEEE VLHCC, pp. 139-143, 2017.

[6] M. Burnett, S. D. Fleming, S. Iqbal, G. Venolia, V. Rajaram, U. Farooq, V.Grigoreanu, and M. Czerwinski, Gender differences and programming environments: Across programming populations, IEEE Symp. Empirical Soft. Eng. and Measurement, Article 28 (10 pages), 2010.

[7] M. Burnett, A. Peters, C. Hill, and N. Elarief, Finding gender inclusiveness software issues with GenderMag: A field investigation, ACM CHI, pp. 2586-2598, 2016.

[8] M. Burnett, S. Stumpf, J. Macbeth, S. Makri, L. Beckwith, I. Kwan, A. Peters, and W. Jernigan, GenderMag: A method for evaluating software's gender inclusiveness. Interacting with Computers 28(6), pp. 760-787, 2016.

[9] P. Cafferata and A. M. Tybout, Gender differences in information processing: a selectivity interpretation, in Cognitive and Affective Responses to Advertising, Lexington Books, 1989.

[10] J. Cao, K. Rector, T. Park, S. Fleming, M. Burnett, and S. Wiedenbeck, A debugging perspective on end-user mashup programming, IEEE VLHCC, pp. 149-159, 2010.

[11] J. Cassell, Genderizing HCI, In The Hand-book of Human-Computer Interaction, M.G. Helander, T.K. Landauer, and P.V. Prabhu (eds.). L. Erlbaum Associates Inc., pp. 402-411, 2002.

[12] S. Chang, V. Kumar, E. Gilbert, and L. Terveen, Specialization, homophily, and gender in a social curation site: findings from Pinterest, ACM CSCW, pp. 674-686, 2014.

[13] G. Charness and U. Gneezy, Strong evidence for gender differences in risk taking, J. Economic Behavior & Organization 83(1), pp. 50–58, 2012.

[14] C. Coursaris, S. Swierenga, and E. Watrall, An empirical investigation of color temperature and gender effects on web aesthetics, J. Usability Studies 3(3), pp. 103-117, May 2008.

[15] S. Cunningham, A. Hinze and D. Nichols, Supporting gender-neutral digital library creation: A case study using the GenderMag Toolkit, Digital Libraries: Knowledge, Information, and Data in an Open Access Society, pp. 45-50, 2016.

[16] T. Dohmen, A. Falk, D. Huffman, U. Sunde, J. Schupp, G. Wagner. Individual risk attitudes: Measurement, determinants, and behavioral consequences, J. European Econ. Assoc. 9(3), pp. 522–550, 2011.

[17] A. Durndell and Z. Haag, Computer self efficacy, computer anxiety, attitudes towards the Internet and reported experience with the Internet, by gender, in an East European sample, Computers in Human Behavior 18, pp. 521–535, 2002.

[18] J. Hallström, H. Elvstrand, and K. Hellberg, Gender and technology in free play in Swedish early childhood education, Int. J. Technology and Design Education 25(2), pp. 137-149, 2015.

[19] K. Hartzel, How self-efficacy and gender issues affect software adoption and use, Commun. ACM 46(9), pp. 167–171, 2003.

[20] C. Hill, M. Haag, A. Oleson, C. Mendez, N. Marsden, A, Sarma, and M. Burnett, Gender-inclusiveness personas vs. stereotyping: Can we have it both ways? ACM CHI, pp.6658-6671, 2017.

[21] W. Hou, M. Kaur, A. Komlodi, W. Lutters, L. Boot, S. Cotten, C. Morrell, A. Ant Ozok, and Z. Tufekci, Girls don't waste time: Pre-adolescent attitudes toward ICT, ACM CHI, pp. 875-880, 2006.

[22] A. Huffman, J. Whetten, and W. Huffman, Using technology in higher education: The influence of gender roles on technology self-efficacy, Computers in Human Behavior 29(4), pp. 1779–1786, 2013.

[23] C. Kelleher, Barriers to programming engagement, Advances in Gender and Education 1, pp. 5-10, 2009.

[24] J. Margolis and A. Fisher, Unlocking the Clubhouse: Women in Computing, MIT Press, 2003.

[25] N. Marsden and M. Haag, Evaluation of GenderMag personas based on persona attributes and persona gender, HCI International 2016 – Posters' Extended Abstracts: Proceedings Part I, pp. 122-127, 2016.

[26] C. Mendez, H. S. Padala, Z. Steine-Hanson, C. Hilderbrand, A. Horvath, C. Hill, L. Simpson, N. Patil, A. Sarma, M. Burnett, Open Source barriers to entry, revisited: A sociotechnical perspective, ACM/IEEE ICSE 2018.

[27] Christopher Mendez, Zoe Steine Hanson, Alannah Oleson, Amber Horvath, Charles Hill, Claudia Hilderbrand, Anita Sarma, Margaret Burnett, Semi-Automating (or not) a Socio-Technical Method for Socio-Technical Systems, IEEE VL/HCC, 2018 (to appear).

[28] J. Meyers-Levy, B. Loken, Revisiting gender differences: What we know and what lies ahead, J. Consumer Psychology 25(1), pp. 129-149, 2015.

[29] J. Meyers-Levy, D. Maheswaran, Exploring differences in males' and females' processing strategies, J. Consumer Research 18, pp. 63–70, 1991.

[30] A. O'Leary-Kelly, B. Hardgrave, V. McKinney, and D. Wilson, The influence of professional identification on the retention of women and racial minorities in the IT workforce, NSF Info. Tech. Workforce & Info. Tech. Res. PI Conf., pp. 65-69, 2004.

[31] Alannah Oleson, Christopher Mendez, Zoe Steine-Hanson, Claudia Hilderbrand, Christopher Perdriau, Margaret Burnett, Andrew Ko, Pedagogical Content Knowledge for Teaching Inclusive Design, Pedagogical Content Knowledge for Teaching Inclusive Design. ACM ICER, 2018, 9 pages (to appear).

[32] Piazza Blog, STEM confidence gap. Retrieved September 24th, 2015, http://blog.piazza.com/stem-confidence-gap/

[33] R. Riedl, M. Hubert, and P. Kenning, Are there neural gender differences in online trust? An fMRI study on the perceived trustworthiness of EBay offers, MIS Quarterly 34(2), pp. 397-428, 2010.

[34] D. Rosner and J. Bean, Learning from IKEA hacking: I'm not one to decoupage a tabletop and call it a day, ACM CHI, pp. 419-422, 2009.

[35] A. Shekhar and N. Marsden. Cognitive Walkthrough of a learning management system with gendered personas. 4th Gender & IT Conference (GenderIT'18), pp. 191-198, 2018. doi:10.1145/3196839.3196869

[36] S. Simon, The impact of culture and gender on web sites: An empirical study, The Data Base for Advances in Information Systems 32, pp. 18-37, 2001.

[37] A. Singh, V. Bhadauria, A. Jain, and A. Gurung, Role of gender, self-efficacy, anxiety and testing formats in learning spreadsheets, Computers in Human Behavior 29(3), pp. 739–746, 2013.

[38] R. Spencer, The streamlined cognitive walkthrough method, working around social constraints encountered in a software development company, ACM CHI, pp. 353-359, 2000.

[39] E. Weber, A. Blais, and N. Betz, A domain-specific risk-attitude scale: Measuring risk perceptions and risk behaviors, J. Behavioral and Decision Making 15, pp. 263-290, 2002.

[40] C. Wharton, J. Rieman, C. Lewis, and P. Polson, The cognitive walkthrough method: A practitioner's guide. In Usability Inspection Methods, pp. 105-140, 1994.

Building socio-technical systems for representing citizens' voices in humanitarian interventions

Luke Church
Africa's Voices Foundation / University of Cambridge
Cambridge, UK
luke@church.name

Rita Zágoni
Africa's Voices Foundation
Nairobi, Kenya
rita@africasvoices.org

Alexander Simpson
Africa's Voices Foundation
Cambridge, UK
Alexander.Simpson@ cl.cam.ac.uk

Sharath Srinivasan
Africa's Voices Foundation
Nairobi, Kenya
sharath@africasvoices.org

Alan Blackwell
University of Cambridge
Cambridge, UK
Alan.Blackwell@cl.cam.ac.uk

***Abstract*—We describe the work of Africa's Voices Foundation (AVF), a charity based in Nairobi and Cambridge, representing citizens' voices in humanitarian interventions in East Africa. AVF addresses a range of topics including education in Kenyan refugee camps, trust in government in Somalia, and health practices associated with disease outbreaks. We describe the methodology AVF have developed for gathering and analysing citizens' opinions. This methodology includes the use of interactive radio to gather opinions via SMS, natural language processing tools for low resource languages, and custom developed software to enhance the effectiveness of researchers. We reflect on the lessons learnt throughout this process, and how they might apply to other domains.**

Keywords— Humane Computing, Social Science, Natural Language Processing

I. Introduction: Interactive Radio at Africa's Voices Foundation

In global humanitarian interventions, the people who have problems do not always find it easy to communicate with the people who hope to solve their problems. The two groups often speak different languages. The people with the worst problems often have low levels of literacy. And in conflict regions, or locations of natural disasters, the communications, transport and administrative infrastructures are likely to be very fragile, if they exist at all. All of these factors make collaborative problem-solving very challenging indeed.

Africa's Voices Foundation is a Non-Governmental Organisation (NGO) based in Nairobi, Kenya and Cambridge, UK. AVF works with partners serving humanitarian needs, such as UNICEF and Oxfam. AVF provides research design, tooling, execution and analysis to help the partners design, monitor and evaluate their programmes in consultation with, or informed by, citizens of the countries where those programmes will operate. In many ways, the work of AVF resembles forms of applied social science that are familiar in commercial applications for high income countries such as market research, opinion surveys, population census, public health assessments and so on. However, in these politically disrupted and low-resource settings, the actual mechanisms of social science research are very different. Furthermore, the assumptions about what a problem-solving outcome might look like will be very different in a multinational aid context than in commercial product development. This allows us to rethink fundamental aspects of applied social science, in the context of new problems, with the ability to also rethink the technical infrastructure that supports such problem-solving.

AVF uses technology, often developed in collaboration with the University of Cambridge, to provide these applied social science services to assist others' problem-solving. This paper explores the design of that technology.

In order to gather the information about citizens' opinions, Africa's Voices have developed a standard methodology for gathering data using interactive radio. During a radio show, the host will ask questions and encourage participants to send SMS messages with answers to a free shortcode. This method has proved effective for gathering citizens' opinions in hard to reach communities, such as informal settlements.

An example project, DREAMS [1] looked at what cultural and gender beliefs prevented girls from staying in secondary schools. After setting out the context in the show of 'a young girl who completed her primary school education and performed well but did not continue to secondary school, without giving her reason', the host asked 'What do you think were the reasons she dropped out of school?'.

Some of the answers that were texted in were read out on air, creating a space for a dialog around a topic. When participants sent an answer, they were sent back a demographic questionnaire and, in some projects, follow on questions requesting further information.

There is some logic that determines how to respond to an SMS, for example by sending out a demographic survey or responding to the message with a different question based on which language a participant wants to use. This logic is expressed in the SMS platform AVF uses, either Echo Mobile and RapidPro. The notation used to express the logic is a visual end-user data-flow programming language, where the flow the user follows - and hence which SMS they receive next - is determined by matching part of their answer with a regular expression.

The data gathered from these questions and surveys is downloaded from the platform and cleaned, for example by removing very short answers to the main questions, unrelated messages, and grouping words with similar meaning together (male, M, man).

After the data has been gathered, it is considered together with existing social theories to build a coding frame. Messages within the dataset are then coded into the scheme using either a spreadsheet or a custom developed tool named Coda[1].

The data is then analysed, and a report prepared, using a mixed method approach of rich messages from individuals

[1] http://www.africasvoices.org/ideas/newsblog/introducing-our-latest-analysis-tool-coda/

and statistical reporting of prevalence. For example, the answers to the questions above were coded according to a frame derived from the Social Ecological Model [2] about whether reasons were from the individual, the home and family, community or society. The report highlights that the most prevalent category of audience beliefs were about family (1042 messages), together with a quoted message:

"She could have dropped out of school due to too many household chores which deprived her of study/school time." Male, 24, Migori

These reports are then delivered and discussed with the partner organisation to assist them in developing programmes for intervention or further research.

II. Designing Socio-Technical Systems for Problem Solving

In gathering and analysing this data, together with the development and dissemination of social theories and supporting the design and evaluation of interventions, AVF are constructing socio-technical artefacts. The successful design and execution of a project includes many steps, some using more computer time (downloading data from an SMS platform), some using more people time (developing a coding scheme) and some that are a mixture (cleaning data/applying a coding scheme).

We suggest that the design and operation of these assemblages of social, technical and data systems represent an important configuration to support human problem solving. AVF's experience in creating them also reveals a number of challenges that may generalise beyond the specific geographic focus of East Africa.

Whilst there has been extensive study within the Visual Languages community of artefacts to support end-user programming, such as spreadsheets and visual programming languages, there is little research on how these artefacts are integrated, together with manual processes, into workflows that have many steps. In large commercial development contexts building online services this separation is supported by the difference between 'development' and 'operations', one responsible for building and evolving systems, the other for maintaining and operating the systems. However, technology has developed sufficiently that an organisation like AVF which has a number of staff who would be considered to be 'end-user programmers' has built technology that needs an 'end-user operations' team that has some of the skills that would typically be associated with a site reliability function [3].

The approach AVF had previously adopted, in common with other organisations engaging in technically supported problem solving, was an ad-hoc assemblage of spreadsheets, regular expressions and python scripts. Whilst this was effective in enabling the organisation to build solutions, there were a number of problems.

The fragmentation of the cleaning and analysis process into multiple different tools makes it very difficult to debug any problems that arise, and requires a great deal of discipline in handling the data files to ensure integrity throughout the analysis process. There have been several occurrences where it has become clear in the analysis that something has gone wrong earlier in the process, but it has been difficult to reproduce the exact steps followed, and consequently time was consumed to locate and fix the problem. This is especially the case with aspects that involve any manual steps, such as labelling.

Delays are especially problematic in the context in which AVF works in two ways: Firstly, the time it takes to complete an analysis determines the kinds of social problems that AVF can effectively engage with. If it takes too long, the opportunity for the partner organisation to intervene in the circumstance can have passed; this is especially the case in crisis circumstances like managing disease outbreaks. Secondly, the funding structures typically associated with the NGO sector only allow funding to be spent directly contributing to a project, making it difficult to build financial reserves to handle project overruns or to invest in reusable components.

With a setup like this where timing is important and there is considerable manual work, the appeal of technologies like machine learning is enormous. Considering just cleaning, if machine learning was successful it would provide AVF with a tool that could learn patterns of how to clean the messages in a given language. In the next project, this would mean that less time would be spent annotating the messages, decreasing cost and the delay before the first results were communicated to the client and allowing the social scientists to focus on research questions.

From 16 projects in 4 years, AVF has learnt a number of lessons, both technical and organisational in deploying end-user programming and machine learning techniques that are both technically and organisationally viable. These lessons may well apply to other small organisations attempting to address human issues with technology.

III. Lessons Learnt

A. *Technology should be used to amplify other problem solving capabilities, not as an end in itself*

As we discuss at length in a parallel submission [4], there is a tendency to design machine learning technologies as a 'solution', along the lines of 'if we can build and train a model that's good enough, it can clean the data'. This approach has not been effective for AVF. The low resource nature of the languages in use combined with the rapidly evolving context has resulted in projects adopting this strategy failing to deliver a sufficiently good model and consequently creating a serious risk to the project.

Focusing instead on the workflows that AVF is trying to solve, and for each project considering how to enable the social scientists to be the most effective by automating repetitive or mundane tasks, often with statistical techniques, has proven a much more effective strategy.

B. *Technical configurations need to encourage sustainable courses of action*

Organisations like AVF operate projects with very compressed timelines, both due to the funding constraints of the NGO sector, and due to the urgency of many of the projects. This results in analysts often taking the quickest available shortcut to get a result, even if over the long term this is not the best outcome.

Considering this as a problem of attention investment [5], the discount value on attention saved in the future compared

to time saved now is very high. However whilst this is an understandable response to the stress of the work, it doesn't necessarily meet the organization's longer term goals, for example being able to reproduce an analysis at any later point in time.

In order to help address this, we've standardised pipelines that connect all of the pieces of a typical AVF project together (downloading SMS data, cleaning it, de-identifying it, packaging it for analysis), into standard templates using Docker containers [6].

As well as being the easiest way of getting started with a project, these templates also provide a level of reproducibility by archiving a copy of all the code that ran, including any 3rd party code, into a reusable image. This enables the solution that was adopted for any particular problem to be archived for later reference by the organisation.

C. *Debugging tools need to span human and technical activities*

It is well understood that defects occur in end user programming contexts, and there has been substantial research on improving development environments to support defect localisation and repair [7]. However, in the socio-technical systems described above, defects can arise from a wider range of places, including problems in the presentation of the radio shows, accidental duplication of files, using obsolete versions of datasets, changes by once-off executed scripts, incorrect labelling by machine learning systems, etc.

Finding issues that occur from combinations of manual and automated processes can be very difficult, as without very strict practices (which themselves cause inefficiency), it can be impossible to reproduce the circumstances causing the problem.

To help address this issue, we've developed an immutable tracing data-structure, that efficiently records the input and output of each step. These steps could be function calls in a script, or they could be manual annotations of a message by a translator. As each piece of data now carries its provenance with it, it is possible to both much more effectively debug issues that arise and link analysis results back to the messages they are grounded in.

D. *Prototype with standard applications, build tools to amplify productivity gains*

Requirements gathering for building custom systems for AVF has proven very difficult. It is difficult to know how to build a tool that will fit into a research process that is also changing to adapt to the problem being solved. A number of attempts to build bespoke tools have resulted in artefacts that in practice haven't been used.

By contrast the technologies that have been adopted at AVF were often built by understanding how the researchers were using existing tools such as a spreadsheet in an unconventional way (for labelling data), and then building a custom application that optimised the user interface and software architecture for that specific task, as we have mentioned for example in Coda.

The standardisation of the pipelines into templates also reflects this, but focusing instead on learning from the scripts that the end-user developers had written rather than commercial applications.

IV. Conclusion

We suggest that AVF offers an example of a new kind of model for using technology to enhance human problem solving, constructing assemblages of social and technical systems to address humanitarian needs. It has shown the need for new methods of designing, developing, debugging and deploying end-user created realtime data systems.

Acknowledgments

Africa's Voices Foundation is supported by The William and Flora Hewlett Foundation, The David and Elaine Potter Foundation and The Cairns Charitable Trust.

The authors would like to thank Ben Laurie for initial conversations around the design of provenance tracing data structures.

References

[1] Africa's Voices Foundation Team, "Listening to Kenyans to understand the obstacles to girls completing secondary school," 1st report for DREAMS project, Sep. 2017.

[2] Centers for Disease Control and Prevention, "The Social-Ecological Model: A Framework for Prevention," *CDC - Centers for Disease Control and Prevention*, 01-Mar-2018. [Online]. Available: https://www.cdc.gov/violenceprevention/overview/social-ecologicalmodel.html. [Accessed: 12-Jul-2018].

[3] B. Beyer, J. Petoff, C. Jones, and N. R. Murphy, *Site Reliability Engineering*, 1 edition. O'Reilly, 2016.

[4] A. Blackwell *et al.*, "Computer says 'don't know' - interacting visually with incomplete AI models," submitted to *Designing Technologies to Support Human Problem Solving - A Workshop in Conjunction with VL/HCC 2018*, Lisbon, Portugal.

[5] A. F. Blackwell, "First steps in programming: A rationale for attention investment models," *Proceedings - IEEE 2002 Symposia on Human Centric Computing Languages and Environments, HCC 2002*, pp. 2–10, 2002.

[6] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014.

[7] A. J. Ko *et al.*, "The state of the art in end-user software engineering," *ACM Computing Surveys*, vol. 43, no. 3, pp. 1–44, 2011.

Position Paper: Broadening Participation in Online Problem Solving by Increasing Awareness of Common Contributor Qualities

Michelle Ichinco
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA, USA
michelle_ichinco@uml.edu

***Abstract*—Collaborative problem solving efforts, like open-source projects, benefit from diversity. Non-expert outsiders may feel unqualified to participate, even if they have unique and valuable skills. An in-person problem solving context might allow contributors to find commonalities beyond expertise that make them feel like they belong, but online systems often do not provide affordances for awareness of contributor qualities and interests. Instead, newcomers can typically view information like number of contributions or length of participation, which can be intimidating. Gender and location may further exclude potential contributors who do not match the existing population. Yet, potential contributors likely have commonalities with existing community members that are not visible in typical problem solving communities. This position paper argues for enabling potential contributors to discover common qualities or interests with experienced contributors in order to encourage more diverse participation in online problem solving communities.**

***Index Terms*—Open source, diversity, newcomers, community**

I. Introduction

Due to the collaborative nature of crowdsourced problem-solving, diversity can have a significant impact on the productivity and output of a group problem solving effort. Studies have shown that spoken language, country of origin, gender, and interest diversity have positive impacts on productivity or market success [2], [4], [10], [12]. Yet, open-source software projects have significant issues with discrimination [9], which may prevent minorities from entering projects or may cause them to hide their identities. Thus, a newcomer with less experience may find both the experience levels and homogeneity of a community intimidating.

Surveys in open source software reveal that many contributors focus on the programming skills and gender of other contributors. A survey of GitHub open-source developers revealed that 98% of respondents were aware of the programming skills of other collaborators and 48.6% were aware of gender [12]. Research on peoples' perceptions of other GitHub members also revealed a focus on experience over other qualities [7]. Only 3% of a survey's respondents were aware of the hobbies and political views of *most* of their other teammates and only 17% were aware of the hobbies and political views of *some* of their other teammates [12]. Online problem solving communities lack the opportunity to share and discover this type of information, especially compared to in-person contexts.

This position paper proposes supporting online problem solving communities, like open-source communities, in sharing the types of personal qualities, interests, and opinions that would naturally arise during in-person teamwork and that lead to stronger community connections. Those working together on a project within a company may discover interests through natural interactions, like discussing weekend activities and their home lives through lunches or happy hours. They will also have some contextual information to find commonalities simply from existing in the same space, like age, style, language, and ethnicity. I propose extracting this type of information from open source community members and making it available, especially to newcomers. I hypothesize that sharing personal information will enable minority newcomers to make critical connections to the community.

II. Past and Existing Initiatives

One way to support diverse populations is to provide mentoring or extra incentives to join a community. For example, the Ada Initiative aimed to support women in open-source communities by providing repositories, policies, and workshops [1]. In a similar vein to open-source, a recent study showed that mentorship improved newcomer contributions to Stack Overflow [6]. Research has also explored how to ensure diversity is expressed through Wikipedia articles, but focuses on the output of the articles, rather than the effect on the community members [5]. Little research addresses ways to support integration of under-represented groups into online problem-solving communities, like open-source communities, especially through sharing interests and personal traits.

III. Theoretical Grounding

Psychology research has addressed the development of theory and measurement of 'sense of community.' One of the predominant models, by McMillan and Chavis, has four elements: membership, influence, integration and fulfillment of needs, and shared emotional connection [8]. This position

paper focuses on increasing the existence and visibility of *shared emotional connection* because it is often missing or invisible to newcomers. McMillan and Chavis define *shared emotional connection* as "the commitment and belief that members have shared and will share history, common places, time together, and similar experiences" [8]. Shared emotional connection does not require that the community members have participated in the shared history together or be in the same place, as long as they identify with a common shared history. Examples of elements of a shared emotional connection include: important events, spiritual bonds, and frequent interaction.

Interaction in community problem-solving efforts typically makes elements like membership, influence, and common goals available, but little attention is paid to emotional connection. Not only is emotional connection missing from the frameworks for community problem solving, but also in the ideals of the community members. In a survey of GitHub contributors, respondents said: "more about the contributions to the code than the 'characteristics' of the person (40, male, N America)" and "code sees no color or gender (34, male, W&NEurope)" [12]. This presents a challenge: some members of open-source communities are disinterested in knowing more about the characteristics of other community members. Yet, women *are* discriminated against in these communities [9], indicating that bias exists, whether explicit or implicit. The goal of the proposed research direction is to increase awareness of similarities between community members primarily to support newcomers, but ideally also to reduce bias through the creation of stronger community ties.

IV. Increasing Awareness of Contributor Similarities in Online Problem-Solving Communities

This work hypothesizes that increasing the shared emotional connection in online problem-solving communities, like open-source projects, will support more diverse populations. Encouraging cultivation of shared emotional connections necessitates exploration of the qualities that community members would be willing to share, as well as how to support the sharing of these qualities.

A. Qualities

Two potential types of qualities that could foster shared emotional connection are: 1) project-independent qualities like interests, hobbies, or backgrounds, and 2) project-related interests.

Project-independent qualities are more likely to be the common basis for shared emotional connections in person. For example, having lived in similar cities with someone or liking the same music can support further discussion and general feelings of connection. Sharing these qualities may be more difficult to facilitate in open-source communities, where users are highly focused on the tasks. Open-source community members are often volunteers, who may not have or want to spend extra time getting to know each other beyond the tasks at hand. Yet, sharing information about contributors' lives, especially big life events like moving cities or the birth of a child, may make others more sympathetic, improving the quality of the community. Contributors may also have privacy concerns with sharing some types of personal information, which may necessitate flexibility in allowing members to share the information they feel comfortable with, rather than mandating that they share specific information, like location. Even with these challenges, discovering shared qualities beyond the projects will likely forge stronger connections than project-related information and enable minorities to feel more connected, even if others from their group are not present.

Open-source projects currently provide contribution-specific user information, but lack space for sharing more general technical interests and skills. Open-source projects typically have discussion, which can reveal some project-related interests and skills, but it may be difficult for new members to integrate themselves in to these discussions. Much of the project-contribution information in open-source communities is available, but less may be available about the user. For example, information about community members' career background, knowledge and skills beyond the current project, and preferences about types of projects or roles could help connect members across length or number of contributions. Open-source contributors may be more comfortable sharing this information and more willing to spend time sharing this information because it is much more related to the project and goals of the community than elements of their personal lives. On the surface, this type of information does not seem like it would support shared emotional connections, but due to the importance of career goals, this type of information may have a similar effect to more personal information.

Research should explore the willingness of community members to share these two types of information. Work should also determine whether discovering either set of shared qualities helps increase shared emotional connections in open-source or other community problem-solving contexts.

B. Sharing Information

If knowledge of shared project-independent or project-related qualities improves community, we need to determine the best ways to elicit member qualities and make that information available to newcomers in ways that make newcomers feel connected.

1) Collect and Present: Collecting and presenting qualities would take a static set of interests or traits and present them to other community members to establish deeper connections. This method requires limited continued effort, but on its own, is less dynamic and likely would not enable interaction between community members about their common traits.

One way to do this would be to encourage sharing qualities through a user profile, like a GitHub user profile. Currently, GitHub has optional profile information: name, bio, URL, company, location, and image. A study showed that in forming impressions of other users, members typically focus on the contributions displayed, including the types of projects,

languages, and discussion, rather than bios or other personal information [7]. Part of the reason for this may be that many users do not share personal information in these areas that would help to make connections.

There are likely a variety of ways to improve the available information about open-source projects. For project-related information, communities like GitHub could automatically extract a summary from a user's contributions. This could include information like languages used, types of projects contributed to, and snippets of communications. These type of communities could also provide more structure to encourage users to include information about themselves, or suggest information to share based on the types of information others have shared. However this information is collected, when viewing a user profile, shared information could be highlighted to help users notice common qualities.

2) Exchange: Enabling users to exchange their traits would likely support more natural communication about similarities, but is problematic for several reasons. Current methods of communication in open-source communities, like issues and forums, should and do focus on the goals of the project. Personal conversations mixed into task-based communication would be problematic and likely very unpopular. While open-source communities can use other informal spaces for communication, like Slack, these present other problems, like leaving out users in non-majority time zones and users who do not have constant internet connection [3], [11]. Furthermore, many busy volunteer contributors may not feel it is important to take part in non-task interactions. If only a subset of contributors take part, it diminishes the potential for newcomers to find commonalities within the community.

V. Research Directions

This position paper hypothesizes that increasing the amount of shared personal information in online problem-solving communities, like open-source projects, will support creation of shared emotional connections between group members. With the abundance of data demonstrating the benefit of and need for different varieties of diversity in problem-solving groups, the communities now need support for including minorities. This research direction raises many questions for future research, such as:

- Which personal qualities will online problem solving community members be willing to share?
- Which personal qualities (if any) are most effective at building shared emotional connections that strengthen newcomer bonds to the communities?
- What are the best ways to obtain this information from community members?
- How can we encourage existing community members, especially those who do not believe discrimination exists within their communities, that these measures are important?
- What are the best ways to share community members' personal qualities with new and existing community members?
- How much or little engagement with shared traits is needed in order to build these emotional connections within online problem-solving communities?
- Can these shared common qualities and the connections formed reduce bias toward minority groups?

References

[1] Ada Initiative.

[2] Jilin Chen, Yuqing Ren, and John Riedl. The effects of diversity on group productivity and member withdrawal in online volunteer groups. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 821–830. ACM, 2010.

[3] Dave Cheney. Why Slack is inappropriate for open source communications | Dave Cheney.

[4] Sherae Daniel, Ritu Agarwal, and Katherine J. Stewart. The effects of diversity in global, distributed collectives: A study of open source project success. *Information Systems Research*, 24(2):312–333, 2013.

[5] Fabian Flck, Denny Vrandei, and Elena Simperl. Towards a diversity-minded Wikipedia. In *Proceedings of the 3rd International Web Science Conference*, page 5. ACM, 2011.

[6] Denae Ford, Kristina Lustig, Jeremy Banks, and Chris Parnin. We Don't Do That Here: How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 608. ACM, 2018.

[7] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. Impression formation in online peer production: activity traces and personal profiles in github. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 117–128. ACM, 2013.

[8] David W. McMillan and David M. Chavis. Sense of community: A definition and theory. *Journal of community psychology*, 14(1):6–23, 1986.

[9] Dawn Nafus. Patches dont have gender: What is not open in open source software. *New Media & Society*, 14(4):669–683, 2012.

[10] Marco Ortu, Giuseppe Destefanis, Steve Counsell, Stephen Swift, Michele Marchesi, and Roberto Tonelli. How diverse is your team? Investigating gender and nationality diversity in GitHub teams. Technical report, PeerJ Preprints, 2016.

[11] Slack. Where work happens.

[12] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3789–3798. ACM, 2015.

UNAKITE: Support Developers for Capturing and Persisting Design Rationales When Solving Problems Using Web Resources

Michael Xieyang Liu[*], Nathan Hahn[*], Angelina Zhou[†], Shaun Burley[‡], Emily Deng[†], Jane Hsieh[§], Brad A. Myers[*], Aniket Kittur[*]
[*†‡]Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[§]Oberlin College, Oberlin, OH, USA
{xieyangl, nhahn, bam, nkittur}@cs.cmu.edu[*], {ajzhou, edeng}@andrew.cmu.edu[†], me@shaunburley.com[‡], jhsieh@oberlin.edu[§]

***Abstract*—UNAKITE is a new system that supports developers in collecting, organizing, consuming, and persisting design rationales while solving problems using web resources. Understanding design rationale has widely been recognized as significant for the success of a software engineering project. However, it is currently both time and labor intensive for little immediate payoff for a developer to generate and embed a useful design rationale in their code. Under this cost structure, there is very little effective tool support to help developers keep track of design rationales. UNAKITE addresses this challenge for some design decisions by changing the cost structure: developers are incentivized to make decisions using UNAKITE's collecting and organizing mechanisms as it makes tracking and deciding between alternatives easier than before; the structure thus generated is automatically embedded in the code as the design rationale when the developer copies sample code into their existing code. In a preliminary usability study, developers found UNAKITE to be usable for capturing design rationales and effective for interpreting the rationale of others.**

Keywords—design rationales, programming support tools, sensemaking

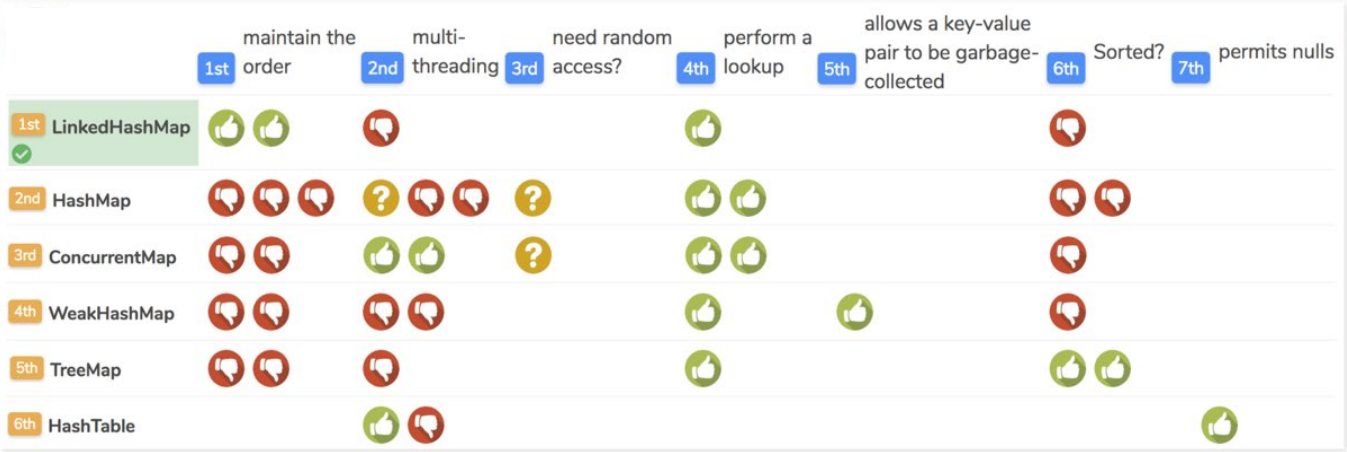

Fig. 1: The *comparison table* that UNAKITE generated as a result of web snippets collected and characterized by a participant duing a usability study of UNAKITE.

REFERENCE

[1] Michael Xieyang Liu, Shaun Burley, Emily Deng, Angelina Zhou, Aniket Kittur, Brad A. Myers, "Supporting Knowledge Acceleration for Programming from a Sensemaking Perspective", *Sensemaking Workshop at CHI'2018 Conference on Human Factors in Computing Systems*, April 21, 2018. https://lxieyang.github.io/assets/files/pubs/kap-sensemaking-workshop/kap-sensemaking-workshop.pdf

Problem Solving and the Future of Computing Position Statement

Dastyni Loksa
The Information School
University of Washington
dloksa@uw.edu

Andrew J. Ko
The Information School
University of Washington
ajko@uw.edu

***Abstract*—In this position paper we discuss explicit problem solving instruction for computer programming, and its possible impacts on the development of programming languages and development tools. Computing education currently provides instruction on programming languages and development tools but provides little to no instruction on the cognitive processes to solve programming problems. This lack instruction is due to a lack of understanding about how successful programmers solve programming problems and we posit that this gap affects computing education, but also hinders the development of new programming languages and tools. We present a framework for programming problem solving as a model of how we might understand and communicate the problem solving process for programming. We then discuss how developing an understanding of the problem solving process has far reaching implications for the development of new programming languages and development tools.**

***Index Terms*—programming, problem solving, cognition**

I. Position Statement

To successfully write working code programmers need to know three things: the programming language and application programming interfaces (APIs) being used, how to use their development tools, and how to solve programming problems. Programming languages, APIs, and development tools are often documented, learnable, and teachable. The process of solving programming problems, however, does not have the same level of documentation. Little is known about the cognitive processes necessary to solve programming problems. Without this knowledge we cannot provide effective documentation or instruction to facilitate the development of important problem solving skills.

The few research studies that have investigated explicit problem solving instruction for programming show the benefits. Bielaczyc et al. investigated the impact of teaching self-explanation, a cognitive strategy of asking and answering questions in an effort to better understand the ongoing mental work that is happening in ones head. They found that students who received explicit training on self-explanation strategies used them more than those without the training and had greater problem solving success [1]. More recently, we found that combining similar explicit training on problem solving, by providing a framework for programming problem solving behaviors (detailed below), promoted greater problem solving success as well as gains in productivity, self-efficacy and avoided the deterioration of growth mindset commonly found in introductory level programming courses [2].

Considering the benefits of explicit problem solving, how might programming instruction differ if our knowledge and documentation for problem solving was as robust as that for programming languages? One step in the direction towards a robust understanding of the problem solving of programmers is offered in our study on novices' self-regulation [3]. In this study we presented a framework that identified five nominally sequential, yet highly iterative, problem solving behaviors that constitute the problem solving process as well as five self-regulation behaviors that mediate the process. The programming behaviors identify categories of actions that a programmer takes while solving problems, while the self-regulation behaviors identify the mental processes that successful programmers engage in to systematically enact the programming behaviors. The framework includes the following programming behaviors:

- Reinterpreting the problem prompt. Programmers must interpret and clarify the problem description for the programming task. As with any problem solving, this understanding is a cognitive representation of the problem used to organize ones' continuing work [4].
- Searching for analogous problems. Programmers draw upon problems they have encountered in the past, either in past programming efforts or even in algorithmic activities from everyday life. By reusing knowledge of related problems, programmers can better conceptualize a problems computational nuances.
- Adapting possible solutions. With some understanding of a problem, programmers seek solutions that will solve the problem by selecting, evaluating, and adapting solutions to related problems or those found in textbooks, online, or from peers and mentors.
- Implementing a solution. With a solution in mind, programmers must translate the solution into code using their chosen programming languages, APIs and tools.
- Evaluating an implemented solution. After implementing a solution, programmers evaluate how their implementation solves the problem by testing and debugging.

The self-regulation behaviors included:

- Planning. Programmers should reflect on what their next

step in a problem solving process should be (e.g., did new information reveal a gap in understanding? What tasks remains for an implementation?) [5]. The more a learner engages in explicit planning, the more successful they should be.

- Process monitoring. Programmers should monitor their progress toward solving a problem (eg., did that change help solve the problem? is this sub-goal complete?, what more needs to be done?). The more learners monitor when a task is complete, the more successful they should be.
- Comprehension monitoring. Programmers should monitor their understanding of computational concepts in problems and solutions (e.g., am I confused? do I really understand how this should function?). The more aware learners are of their misconceptions or gapes in knowledge, the more successful they should be at addressing them.
- Reflection on cognition. Programmers should make judgments about the qualities and limitations of their memory and reasoning (e.g., am I forgetting something? Am I making any assumptions?). The more aware learners are of their cognitive biases, the more likely they are to correct for them.
- Self-explanation. Programmers should explain to themselves why they have come to a conclusion or decision, and then use that rationale to monitor their progress (e.g., this was the right loop condition because it halts at the end of the list). The more learners engage in self-explanation, the more they will find flaws in their reasoning.

This framework is part of our ongoing efforts to understand, document, and teach the problem solving skills involved in computer programming. Some form of this framework has been shown to better support the teaching and learning of programming [2] and useful in describing the problem solving skills of novices [3], but it is one of many possible representations of programming problem solving. While not comprehensive, it serves as a first model of how we might understand and communicate the problem solving processes, providing insights into how programming languages and tools might better support programmers' problem solving.

II. Implications

Developing a robust understanding of programming problem solving could change the landscape of programming tools and methods. What if we had documentation for the problem solving process that was as robust as the documentation for programming languages? The methods of problem solving instruction could be incorporated into intelligent tutors and assessment tools making them more productive learning tools. Leveraging a robust knowledge of problem solving to build programming tools and IDEs that support a programmers cognitive work may significantly reduce the mental work necessary to work with even large code bases. As an example, what might an IDE that prompts a user for a formalized version of *process monitoring* or *planning* look like? Such an IDE could, for instance, support *process monitoring* by identifying the problem solving behaviors listed in the framework above and allow, or even prompt for, the user to track their progress through those behaviors. Another example is an IDE focused on supporting planning and reinterpreting the problem prompt by providing a planning tool where the user can recursively identify the tasks and sub-tasks that need to be achieved. It could allow for the definition of sub-tasks at increasingly finer levels detail, finally converging on the lines of code that solve the sub-task. With the code for each sub-task written, the IDE, and/or future programming languages, could then support the writing of any connective code to form a working program.

The adoption and formalization of problem solving instruction could also have many implications for the advent of new programming languages. Programming languages designed to work with the problem solving processes programmers are trained to engage in could revolutionize the way we currently think of programming, changing the way we conceptualize and represent programming problems, sub-problems, and relationships between software components. It might change the way we think about and design software and lead to yet more expressive languages. For instance, a programming language developed to incorporate and enhance formalized strategy use may reduce the cognitive load of a programming task by automating demanding work or identify next steps or important considerations of the formal strategy being used. No matter what form our understanding of programming problem solving ultimately takes, explicitly teaching programmers to be more aware of their mental processes, strategic in their thinking, and explicit in their actions opens up an exciting range of possibilities for the future of programming, and programming languages.

References

[1] Bielaczyc, K., Pirolli, P. L., Brown, A. L. (1995). Training in self-explanation and self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem solving. Cognition and instruction, 13(2), 221-252.

[2] Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., Burnett, M. M. (2016, May). Programming, problem solving, and self-awareness: effects of explicit guidance. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 1449-1461). ACM.

[3] Loksa, D., Ko, A. J. (2016, August). The role of self-regulation in programming problem solving process and success. In Proceedings of the 2016 ACM Conference on International Computing Education Research (pp. 83-91). ACM.

[4] Falkner, K., Szabo, C., Vivian, R., Falkner, N. (2015, May). Evolution of software development strategies. In Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on (Vol. 2, pp. 243-252). IEEE.

[5] Soloway, E., Spohrer, J. C. (2013). Studying the novice programmer. Psychology Press.

Incremental Understanding and Coordination of Software Re-architecting

Elham Moazzen, Robert J. Walker, Jörg Denzinger, Lora Oehlberg
Department of Computer Science
University of Calgary
Calgary, Canada
Email: {emoazzen, walker, denzinge, lora.oehlberg}@ucalgary.ca

***Abstract*—Real-world software undergoes constant change to keep up with the changing environment around it. Non-trivial software must possess a *software architecture*: a division into smaller pieces, how those pieces are meant to interact, and how those pieces are deployed physically. As a software architecture can have a significant impact on important properties of the software, the architecture for a software system may need to change as the system itself undergoes change, *software re-architecting*. Our recent work examined how software re-architecting proceeds in practice, identifying various problems—particularly the *human aspects* of communicating, discussing, and remembering voluminous details involving incomplete knowledge—that arise that tend to lead its frequent failure. Real re-architecting tends to be organized around small "change steps" that are discussed abstractly, added/modified/elaborated/discarded, and organized in order to achieve the overall goal of an architectural change. The lack of a systematic approach to the communication and record management of change steps, led us to a set of design guidelines for future collaboration tools tailored for re-architecting. Here, we propose a knowledge representation framework for the change process in asynchronous collaboration, without being excessively burdensome nor requiring the presence of complete information not yet available. This framework is a first step toward a re-architecting collaboration tool that would help to systematize the change process without disrupting it. We performed a paper prototype study of this framework to determine if the investment required for constructing, deploying, and evaluating a concrete tool would be warranted. These initial results suggest that the knowledge representation framework supported by the prototype allows software engineers to better present and manage architectural changes relative to traditional mechanisms.**

***Keywords*-Software re-architecting, human aspects, incomplete knowledge, incremental approach, knowledge representation framework, paper prototype, human subjects study.**

I. Introduction

Real-world software undergoes constant change: to fix bugs; to extend functionality; to interact with the changing "ecosystem" around it; and to make internal improvements (e.g., to improve its understandability by developers) [1], [2]. Non-trivial software must possess a *software architecture* [3], [4]: a division into smaller pieces; how those pieces are meant to interact; and how those pieces are deployed physically (e.g., on a specific set of separate computers connected via an intranetwork on which a specific communication protocol is used). As a software architecture can have a significant impact on important properties of the software (such as performance, usability, scalability, etc.), the architecture for a software system may need to change as the system itself undergoes change: this is *software re-architecting*.

Software re-architecting is an important activity in large-scale software systems, where it is not justifiable to rewrite a codebase from scratch. The solutions that exist focus on (formal) technical aspects involved in software re-architecting, which are important, but ignore the human aspects of the phenomenon. Software re-architecting is poorly understood at a practical level. It is widely believed that it is hard to accomplish (e.g., [5], [6]), with potentially devastating effects on the quality of the software system [7].

In our recent work [8], we sought to identify the problems encountered in practice while re-architecting. First, we performed a case study on a known, successful re-architecting: the creation of the "Rich Client Platform" (RCP) of the Eclipse integrated development environment, for client applications in which the standard graphical user interface (i.e., the Eclipse workbench) is inappropriate or too resource-intensive; the case was unusual in that initial plans and detailed records of discussions were available for analysis and comparison with the end result. Second, we interviewed software developers from various organizations who had experience (often negative) in performing re-architecting. In both cases, we found that the key difficulties centred on: (1) necessarily incomplete information until the re-architecting is successfully finished; (2) the need for collaboration between developers, architects, and managers, each with differing expertise and varying knowledge beyond that expertise; (3) additional confusion and errors in connection with ambiguous human language; and (4) difficulties in recording, organizing, and adjusting portions of the re-architecting (plan) as the team's understanding deepens and decisions are revised. More specifically we developed a set of design guidelines derived from our analyses.

Based on these design guidelines, we present a novel knowledge representation (KR) framework for concrete tools aimed at addressing re-architecting, supporting collaboration, incrementality, and iteration. Strong evaluation of the KR framework would require its reification in industrial-strength tools, deployment in the field, and long-term assessment at their effectiveness; at this early stage, we must content ourselves with *in-laboratorio* study of prototypes. As such, we provide a brief overview of a study we performed with eight human participants; we compared a paper prototype based on

the primary aspects of the KR framework, against a traditional set of software development tools and recording the details in email, as per common industrial practice.

The remainder of the paper is structured as follows. Section II provides background on the design guidelines that we derived from our previous studies. Section III describes our KR framework for this problem. Section IV outlines our paper prototype evaluation of the framework. Section V discusses remaining issues. Section VI highlights related work.

II. Background

From our retrospective analysis on the Eclipse RCP re-architecting and our analysis of the interviews we performed with developers who experienced re-architecting, we derived five design guidelines to follow in developing support for software re-architecting.

DG1: Software engineers need a systematic but efficient approach to the communication and record management of the whole process of change.

DG2: Software engineers need a clear understanding about the greater architecture of a system: what the entire code does and how their code affects or is affected by other code.

DG3: Software engineers need to clearly articulate to their collaborators changes that are happening with respect to the greater architecture of the system, especially emphasizing the differences between suggestions. They need support to visually communicate changes with their collaborators to help them externalize their mental model regarding what is going on.

DG4: Software engineers need to record detailed changes together with the communications about those changes and to retrieve information from these records.

DG5: Software engineers need to use state-of-the-art tools for analysis and reasoning to inform their decisions.

The *change process* is the activity involving one or more people to determine the appropriate complex transformation to be performed on the system. The transformation is typically broken down into smaller *change steps* and people tend to talk about individual change steps and the order which they should be applied. Our studies suggest that software engineers need a *systematic approach* to communication and record management in the change process of re-architecting that does not become burdensome to use (i.e., it is "efficient").

III. A Knowledge Representation Framework for Software Re-architecting

We need to structure the way engineers exchange information about the software system throughout the change process; without a common language for representing that knowledge, confusion easily arises. We present a specific interpretation of a solution for this problem: a knowledge representation (KR) framework [9] that captures and represents diverse kinds of information in the change process via lightweight modelling, visualization, and collaboration of stakeholders, enabling the use of human reasoning and analysis tools to inform awareness and decision making. This framework connects all persons who are involved in a re-architecting and connects all the scattered information as a coherent body of knowledge that dynamically evolves as everyone contributes to it.

A. Requirements

We refine the design guidelines into a set of seven requirements on the KR framework, as follows.

R1: The KR framework must communicate the high level architecture of a software system. [from DG1 and DG2]

R2: The KR framework must communicate where changes are happening with respect to the high level architecture of a software system. A change model is iteratively built into it by the collaborators. [from DG1 and DG3]

R3: The KR framework must communicate the order of changes for the sake of considering their priority and dependency. [from DG1 and DG3]

R4: The KR framework must support incomplete change models, as incomplete or inaccurate information about the software system is to be expected. [from DG1, DG2, and DG3]

R5: The KR framework must factor in analysis that is important for the plan stakeholders, such as analyzing completeness of a change model and consequences of decisions for engineers, and planning a sequence of changes for managers. [from DG5]

R6: The KR framework needs to visualize information in different ways for different stakeholders with varying expertise, including those without technical knowledge, without extensive training. [from DG1, DG3, and DG4]

R7: The KR framework must involve fundamental version control features (i.e., branching and merging) to support revision and alternatives. It must also help collaborators to identify and to combine common steps/features across change models (e.g., to borrow from each other). [from DG1 and DG3]

B. Example

In this framework engineers first communicate their understanding of the existing software system with an intuitive, lightweight sketching approach. By iteratively refining this model, they provide "just enough" details in their model of the existing software system and identify ambiguities and answer each others' questions. Figure 1 shows an example involving the help subsystem of Eclipse. The main elements of a sketch diagram are boxes and arrows. They are very simple as if engineers are using pen and paper or drawing on whiteboard. Resulting box and arrow sketches are very informal. They are partial models, representing only any physical (e.g., file and directory) or logical (e.g., methods and classes) parts of the structure of the existing software system that engineers think are relevant to their re-architecting task.

1) Some boxes map directly to a physical unit of source code. For example, IHelp in the APIs box is a file in the source code repository that is a Java interface, while WorkbenchHelp in the ui plugin box is another file in

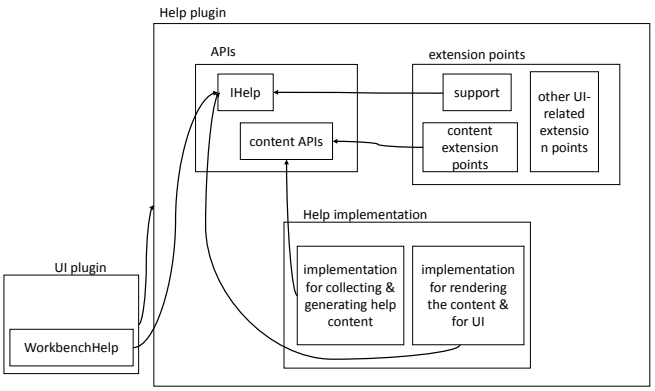

Fig. 1. Sketch of the original help subsystem.

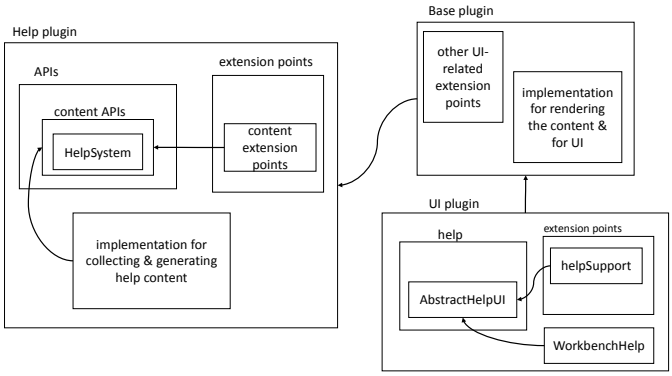

Fig. 2. Sketch of the target, re-architected help subsystem.

the project repository that is a Java class. Engineers can import any physical or logical unit of code and configuration from the project repository into their model. They visualize it with a box if it is a functionality concept or with an arrow if it is a dependency concept in the software implementation.

2) Strictly speaking, the logical units are not limited to programming language entities, e.g., a method in Java; they can even be just a few lines of code if desired. Engineers visualize that concept with a box or arrow. For example, support is a piece of configuration declared in an XML file in the project repository. However, it has an important role in the architecture of the help subsystem, called an extension point in the Eclipse plugin architecture, and it is relevant to the re-architecting thus it is sketched.
3) Some boxes do not directly map to a physical unit in the project repository. For example, the help repository does not include an APIs package or a package called extension points; similarly with content extension points and other UI−related extension points, plus help implementation and the two boxes within it. These are very high-level concepts useful to the engineers during the re-architecting task at hand, not necessarily corresponding to a well-modularized unit in the project repository; each box could ultimately map to hundreds or even thousands of lines of code and configuration that are spread out in different files and directories in the source code repository.
4) The existing system might have other facets that engineers do not care about in their re-architecting task at hand; thus they do not show them in their sketches. They are unimportant context, as is the rest of the Eclipse Platform in this example.

Engineers iteratively refine the high-level sketch to reach the level of detail and accuracy that is necessary for assessing and planning the change (in this example, we assume we are at the point where engineers think that this is enough to proceed for now). They now proceed to lay out a sequence of *change steps* to transform the original architecture into the target architecture. Figures 3 and 4 each shows a static view of a single change step; they are interconnected by the transformations shown in Figure 5. Red boxes and arrows indicate deleted entities and relationships; green, added ones; yellow modified ones. Thus we can see, for example, that IHelp is apparently deleted from the APIs box within the Help plugin box (Figure 3) while HelpSystem is apparently added to the content APIs box within the APIs box within the Help plugin box, and AbstractHelpUI is apparently added to the help box within the UI plugin box (Figure 3); however, from the transformations shown in Figure 5, we can see that IHelp was actually split into HelpSystem and AbstractHelpUI either during or before the latter two were moved to their destination boxes. We can see then that this change step is really a compound transformation of split/move; simpler, primitive transformations could be modelled if desired. Figure 6 illustrates the idea that individual change steps are then sequenced to build up a complete transformation from the original system to the re-architected one.

The framework does *not infer* these little changes that are part of the big change step; it simply records engineers' decisions through interaction with the sketched model.

C. The Framework Dimensions and Core Features

On the sketched model of the existing software system, engineers follow an iterative, three step process: (1) define a change step; (2) commit a change step; and (3) revise a change step, going to Step 2.

The KR framework has five knowledge representation dimensions, representing different information about the change:

1) *Component.* The component dimension represents the logical units of implementation; its design supports R1 and R6. In the KR framework, software developers graphically represent a component with a box. A component can be mapped to its implementation in the code repository. This mapping is expressed by the software developer. The box may not exist in the code repository as an IDE resource or programming language element but it may be realized as chunks in various places in the codebase. The mapping between a component and its implementation in the code repository ranges from

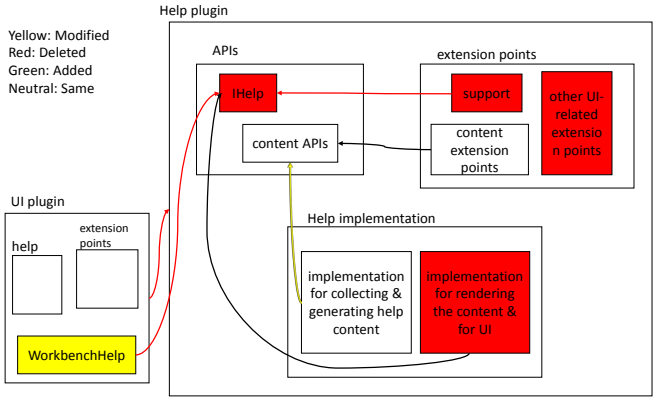

Fig. 3. Change step view: Apparently eliminated concepts.

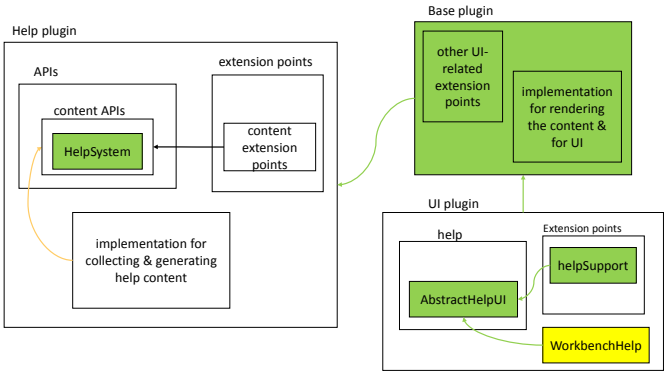

Fig. 4. Change step view: Apparently added concepts.

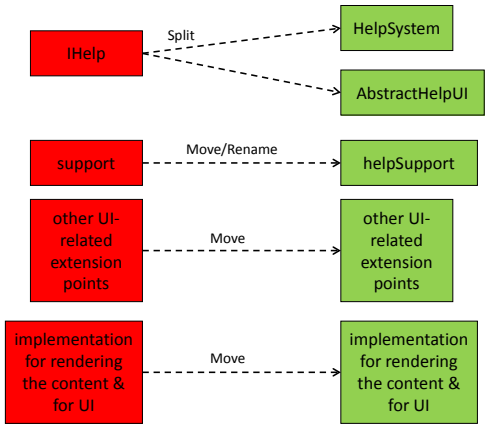

Fig. 5. The transformation relation.

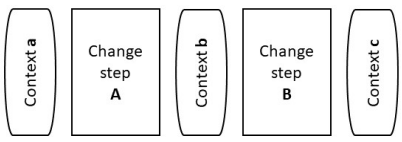

Fig. 6. Sequencing change steps to take an original software system (Context a) to a re-architected system (Context c).

no mapping expressed at all to a complete correspondence to all code chunks and configuration data that implements that component. Software developers can also express other knowledge they might have about the component of interest in the form of natural language content. For example, what they think the purpose of a component is or what assumptions the component makes about other components.

Basic operators to sketch components and work with them in the framework include:

- Add / Delete / Modify a box sketch
- Move / Resize a box sketch
- Add / Delete / Modify a mapping from a box sketch to the code repository
- Add / Delete / Modify an abstraction hierarchy
- Move / Navigate an abstraction hierarchy
- Add / Delete / Modify natural language content
- Annotate natural language content as certain, uncertain, or question

2) *Dependency.* The dependency dimension represents how the individual components collaborate with each other to provide a bigger functionality or service in the system. Its design supports R1 and R6. The dependency dimension ranges from the different types of dependencies that the programming languages and technologies define and that the IDE supports between resources, at one end of the spectrum, to the high-level dependencies that software developers use to communicate and reason about the interaction and collaboration of the components of the existing system, at the other. In this framework, software developers graphically represent a dependency with an arrow. Similar to a component, a dependency can be mapped to pieces of code and configuration which a developer thinks implement that dependency in the code repository. The mapping between a dependency and its implementation in the code repository ranges from no mapping at all to a complete correspondence to anything in the code repository that implements the dependency. Software developers can also express other knowledge they have about the dependency of interest by adding natural language content to the arrow sketch. The natural language content can be certain knowledge, uncertain knowledge, or a question.

Basic operators to sketch dependencies and work with them in the framework may include:

- Add / Delete / Modify an arrow sketch
- Move / Resize an arrow sketch
- Add / Delete / Modify a mapping from an arrow sketch to the code repository
- Add / Delete / Modify an abstraction hierarchy
- Move / Navigate an abstraction hierarchy
- Add / Delete / Modify natural language content
- Annotate natural language content as certain, uncertain, or question

3) *Change step.* The change step dimension is the core

of this framework. This dimension of the framework represents information about the system when engineers investigate and plan an architectural change in the system. Its design supports R2, R3, R4, R5, and R6. This dimension includes:

- Current understanding. It is a "good" understanding of the existing system, represented by high-level boxes and arrows sketches, with mappings to the code repository. "Good" is a subjective measure here and it is decided by software engineers when their understanding of the existing system is complete and detailed enough to investigate and plan changes.
- New understanding. It is what software engineers think the new structure of the existing system would look like conceptually, plus explanations about the new code to create. The "new" is not necessarily the final, target design. It is what they envision the existing system to become next.
- A transition from the current to the new understanding. It is a visual, interactive specification of how the current understanding has to change to reach the new understanding, plus instructions on what changes to make in the code repository.

The change step dimension in this framework ranges from very high-level box and arrow concepts—before and after the change and what changes occur in between—at one end of the spectrum; to more detailed box-and-arrow concepts down the abstraction hierarchy; to eventually the programming language elements and IDE resources in the code repository and explanation of what changes should occur to them, at the other end of the spectrum. It is up to the software engineers where in this spectrum to define a change step. A change step may be revised later on during the process when (1) software engineers' current understanding of the existing system evolves over time or (2) they realize they have to re-architect differently.

The change step dimension includes the following groups of operators:

a) Define change step. This group of operators switches the mode of the framework from collaboration on the conceptualization of the existing system to collaboration on assessing and planning changes on the represented concepts. It consists of a set of basic operators to define changes, including:

- Add / Delete / Split a component
- Add / Delete a dependency
- Keep / Copy / Move a component
- Keep / Copy / Move a dependency
- Rename a component or dependency
- Mark impact (of any of these changes)
- Add / Delete / Modify natural language content
- Add / Delete / Modify code snippet content
- Annotate natural language content as certain, uncertain, or question

b) Resume change step. Software engineers resume working on an interrupted change step. The same perspective as above is activated for them in order to pick up where they left off at the interruption.

c) Commit change step. When software engineers feel a change step is complete enough and detailed enough for them to proceed (to implementation or to the discussion of the next change step), they can commit it. The framework keeps track of all change steps that engineers create in a re-architecting process and revisions to them.

d) View/Modify change step. This group of operators helps software engineers to understand or review a change step that is already completely or partly defined. It represents the following types of information: (1) what has not changed in the transition from current understanding of the existing system to the new understanding of what it should become, (2) what concepts existed in the system before the transition and do no longer exist in the system after the transition, (3) what concepts did not exist in the system before and do exist in the system after the transition, and (4) what interactions engineers did with the framework to define the change step. Item (1) is necessary because it presents those parts of the system that despite being unchanged are pertinent to the change; in other words, they are part of the *important context* for the change. The framework does not clog the clarity of the change step by unimportant context.

4) *Alternative dimension.* The alternative dimension in this framework represents (1) alternative understandings of the existing software system or (2) alternative change steps to it. The framework supports models in parallel that either compete or that lead to multiple concrete solutions. Its design supports R7. The alternative dimension includes basic operators such as:

- **Create an alternative.** When a software engineer wants to express her understanding or her idea of a change that is different from another one's, the framework supports her by explicit expression of alternatives as if she is defining them in the first place, as explained before, giving her operators she needs. The framework keeps track of alternative sketches of the existing system as well as its alternative change steps. These are the knowledge that later on in the process, engineers may feel the need to go back to and reconsider them.
- **Delete an alternative.** Software developers may want to delete an alternative altogether from the framework despite the fact that it would not hurt to just let it be there for potential future reference back to it.
- **Compare alternatives.** After at least one alternative is created for the high-level boxes and arrows model of the existing system or for a change step, software engineers would like to see the difference and discuss the pros and cons of each alternative. This

operator allows them to go back and forth among defined alternatives and supports comparing and contrasting two or more alternatives very directly.
- **Report consensus.** After an attempt to reach a consensus, engineers mark one among all the alternatives and proceed. In other words, it is known in the framework which model is the consensus one.

5) *Time dimension.* The time dimension in this framework simply represents how the knowledge that is managed by other dimensions of the framework evolves. Because as engineers proceed, their understanding of the system/problem is constantly evolving; the steps that they are going to take are constantly evolving; and the desiring goal is constantly evolving. This dimension captures the knowledge of such evolution; its design supports R7. The framework includes a repository for all the models and a version control mechanism that keeps track of (1) the high-level boxes and arrows sketches of the existing system and alternatives, (2) the changes steps and alternatives, and (3) the high-level boxes and arrows sketches of the desired system and alternatives. The basic version control operators to consider include:
   - Commit / Checkout a box and arrow sketch to/from the repository
   - Commit / Checkout a change step to the repository
   - View differences between two versions of a sketch or a change step

It is important to note that the framework *does not* make decisions for software engineers, although it could be enhanced to recommend certain small modifications or to point out specific cases of incompleteness or inconsistency. But in general, understanding of what is needed to be achieved and how to achieve it has to come from people. The framework assists in this endeavour so they do not forget details or fail to have a common basis for discussion.

D. A User Interface Supporting the Framework

We consider one concrete model for the user interface of a tool supporting the KR framework, consisting of: a "before" model of the software system; an "after" model of the software system; a change step model comprising the before and after models and interrelating them with a set of transformations strictly from the before to the after. When specifying a change step model to operate directly on the existing system, the before model could be exactly a model of the existing system; otherwise, each before model is identical to some other after model from another change step model, or the system abstracted by a before model needs to be the same as the system abstracted by some after model. Change step models can thus be organized in a directed acyclic graph, and the interface has to enforce the acyclicity; the graph can have parallel paths and multiple leaves for alternative results.

Model operations allow the user to change the before or after models; there are operation specifications that connect the before to the after; and there are view operations that do not affect the underlying model. Every model entity, operation, and arbitrary locations in a view can be annotated with conversations between team members. Thus, the views themselves must be preserved in order to maintain the context of any extant view-anchored conversation.

The user interface enables a user to create the before model, then copies it to the after model which she can then modify as needed. Standard model-impacting operations on the after model consist of add-, move-, and delete-entity or -relationship. Changes to the before model cannot be model-impacting. View-impacting changes can occur to both models: view-add-, view-move-, view-delete-, view-expand-, and view-compact-entity or -relationship. The user interface can animate the sequence of change steps either for the either change step or individual portions thereof, or can show both the before and after models with transformation arrows.

IV. Preliminary Evaluation of the Framework

Although the requirements for the KR framework were ultimately derived from real world information, there is a risk that we misinterpreted statements, that we over- or under-estimated their importance, or that the real world contexts that we studied were not representative of more general cases. Thus, prior to investment in developing concrete tools and conducting longitudinal studies of their use, we seek initial evidence whether the KR framework appears to address the problems of software re-architecting we had previously identified; as such we conducted a small user study.

A. Study Subject

We sampled an actual re-architecting of a software system that we had developed for teaching purposes: a simulation of a vending machine (SVM). The original version of the SVM was highly monolithic and procedural in nature, making it difficult to extend, test, and debug (see Figure 7); the re-architected version (see Figure 8) was object-oriented, heavily using the design patterns Observer, Singleton, and Facade [10] to reduce coupling and increase cohesion. We selected two parts of this re-architecting that were isolated from each other, using each as an experimental task: (1) separate the functionality of handling coins from the code equivalent to the old VendingMachineLogic box into code equivalent to the new funds box, coordinated by code equivalent to the business logic box; (2) separate the functionality of product management and dispensing from the code equivalent to the old VendingMachineLogic box into code equivalent to the new product box, coordinated by code equivalent to the business logic box. The portions of the business logic in the real solution that implemented these two functionalities did not overlap.

B. The Paper Prototype

We used paper, sticky notes, and string to simulate the user interface of a tool realizing the KR framework. We enlisted the aid of a confederate to play the role of this tool on which it ran, responding to verbal requests of each participant; we trained the confederate as to the set of acceptable commands (e.g.,

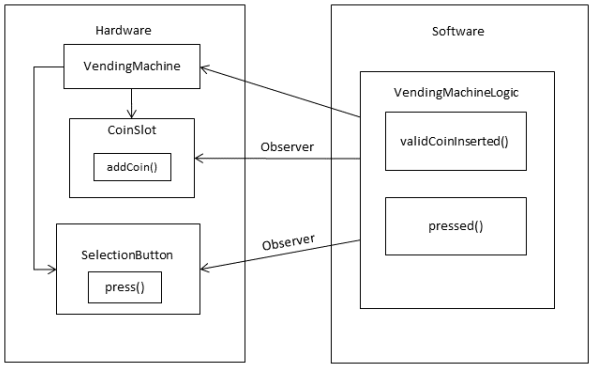

Fig. 7. A high-level view of part of the SVM architecture.

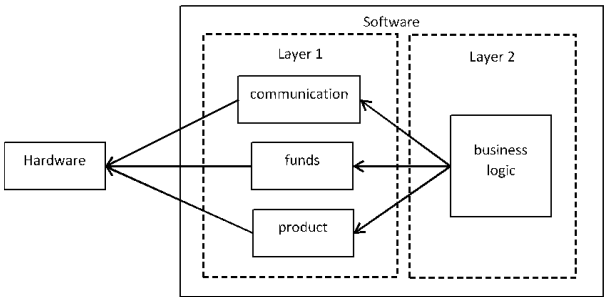

Fig. 8. The new architecture for the SVM.

add box, resize box, move relationship) corresponding to those supported by our user interface design, and the appropriate actions to take in response to each. The setting of the paper prototype can be seen in Figure 9. Participants were given both written notes on the SVM and documentation on interacting with the paper prototype.

C. Study Design

We divided the study into four *study blocks* via a 2×2 factorial design, in which one dimension varied the task order and the other dimension varied the treatment used on the tasks.

Fig. 9. The setting of the paper prototype.

Block 1 involved the application of traditional understanding and communication tools to Task 1, and the application of the paper prototype to Task 2, in that order. Block 2 used the traditional treatment for Task 2 then the paper prototype treatment for Task 1. Blocks 3 and 4 flipped the treatment order but not the task order from Blocks 1 and 2 respectively.

D. Participants

We recruited eight graduate students with industrial software development experience, randomly assigning two of these to each of the four study blocks, where each worked completely independently from all the other participants; there were three people in the room during each session: the experimenter, her confederate simulating the software, and the participant. The participants had not encountered the SVM previously.

E. Protocol

Participants were first given general training on the SVM system and the overall purpose of the re-architecting; they were asked questions about the SVM requiring the use of traditional understanding tools to answer. Participants were given the tasks as described above, in which the point was not to alter the code base but to communicate a plan for performing the task (ostensibly to their manager). We asked the participants to think aloud, and we interrupted to clarify what they were doing and why when it was not self-evident. To simulate "pushback" from a manager in both treatments, we asked for clarification in ambiguous parts of their plan or for correction in parts that we deemed not to have met the goal. Just prior to starting the task in which the paper prototype treatment was to be used, additional training was performed in the use of the paper prototype.

At the end of the study session, participants were asked to fill out a short questionnaire (four questions) asking for their relative opinions on the treatments, on a five-point Likert scale. Detailed field notes were recorded by the experimenter during each session, recording key actions and comments made by the participant and any questions asked and answered.

F. Observations

Two participants strongly preferred the prototype approach, five somewhat preferred it, and only one somewhat preferred the traditional approach. Seven participants strongly preferred the prototype approach for "seeing the big picture"; one preferred it somewhat. All eight participants strongly preferred the prototype approach for ease of plan revisions and ease of iterative communication.

In the traditional treatment, participants universally sketched out a diagram of the new design on paper for themselves, or wrote some pseudocode for the new design in a text editor or on paper. When we reminded participants to communicate their plan via an email to their imaginary manager, they were reluctant, once they had mapped out the changes for themselves. For example: *"I am frustrated because I have to explain changes in language like how different components are behaving and communicating. It is really hard to explain the*

sequence of steps to happen. I am very good at doing this in my brain but explaining the coordination to my manager, that is documenting it, is very hard for me. I solved this in 10 mins. in my head but then deciding how to communicate that plan to someone else took me a lot of time [another 25 mins.]."

All participants announced that the idea of what the prototype approach does is very simple and clear to them. *"It is a nice concept. It can be like a GitHub for architecture."* But applying it for real was less easy at first for some of them.

Issues that participants raised in favour of the prototype approach: (1) visual, aiding understanding and analysis (e.g., *"I liked the prototype approach because it helped me view the weakness of the [new] architecture, the one I built using the provided instructions, which I missed in the manual approach."*); (2) retains best part of traditional approach (e.g., *"The feature of adding notes to changes makes the [prototype] get the good part of [the traditional] approach by explaining the changes explicitly combining it with visualization."*); (3) improves communication (e.g., *"[The prototype approach] makes you present the changes better to the team."*); (4) reduces risk of overlooking something important (e.g., *"[It] makes you consider more details."*); and (5) ease of use (e.g., *"After I understood the prototype, it was easy to use."*).

Specific criticisms of the traditional approach included: (1) tedium (e.g., *"[The traditional] approach takes lots of time to explain what changes need to be done; very tedious."*); (2) loss of detail (e.g., *"[the traditional] approach ... loses a lot of details that will require back and forth communication to clarify. [The prototype approach] makes you consider more."*)

Specific criticisms of the prototype approach included: (1) lack of familiarity (e.g., *"The [traditional] approach is the only way I usually use. Using the tool was a little bit different. It will require time to be familiar with. That is why I tend to give a somewhat preferred to it. It will require some learning curve to be adopted in organizations so everyone understands."*); (2) support for automated warnings (e.g., *"It would be nice if the future version of the prototype tells whenever you made a change that causes compilation error or tests to fail."*); and (3) support in sequencing change steps (e.g., *"In case time/sequence was available, it would help me explain it to my manager. It would also help me remember the reason I decided to do some changes over time."*).

Comments connected to the experiment itself: (1) *"I like the experiment because the code is not complicated to understand but it was still complicated enough to make me think of how I could restructure it. Due to the entanglement of its parts and the dependencies between them it has complexity and I needed to keep the same final behaviour for the user in the new architecture (i.e., preserving the same user experience)."*; (2) *"Since we started with [prototype] experiment, it took more time because getting used to the architecture needs some time which was included in the first step [i.e., the first treatment]. The only reason that manual approach was short was because I had a more clear mind to re-architect the system, as I [had] spent some time with it during the first experiment."*

V. Discussion

It is important to recognize that this was a preliminary study, and so many issues can be pointed to that will need to be addressed in moving ahead.

A. Study Limitations

As an initial evaluation of the concept, we tested only the capability of modelling one possible change step in the prototype of the knowledge representation framework, involving requirements R1, R2, R4, and R6. The other requirements involved multiple change steps or collaboration, demanding longitudinal study we deemed premature at this stage of the research. Thus, we cannot claim that fulfilling those requirements will suffice to solve the problems of re-architecting.

Although we noted the time taken for each task, we do not report it as it is not especially meaningful, given the simulation nature and our interruptions. Participants took roughly a half hour to complete each task. We did note that the second task performed was typically performed in less time regardless of treatment. Whether this represented a true learning effect about the tasks or an effect of learning about our interruptions and questions, is unclear, but a point of concern moving forward.

B. The Paper Prototype Limitations

We tested core concepts of the KR framework and not a user interface. Despite attempts at minimizing investigator bias by training a confederate to simulate a user interface, he was obviously a human being who did not interact with paper, pins, and string as quickly as a software tool would alter a visualization of a change step model. Some participants expressed a certain degree of frustration with this aspect, as it reduced the fidelity of the simulation and demanded mental agility and imagination from the participants. Despite that, we received useful feedback from participants regarding the utility of the concepts.

C. Bias in the Questionnaire

In an attempt to simplify the questionnaire, bias likely crept into the questions being put to the participants. Coupled with the fact that they were graduate students alongside the first author, they likely wished to please. However, we could see from direct observation and the resulting artifacts that they were able to express their ideas for the re-architecting more effectively than in their textual descriptions. Personal experience during dry runs of the study pointed to the same issue: trying to express the notion of a transformation of a poorly encapsulated segmented of code is painful.

D. Potential Transfer of Lessons Learned to Other Problems

Although our KR framework is obviously geared to technically-literate users working in a narrow domain (i.e., software developers and associated stakeholders performing software re-architecting), we are strongly of the opinion that the core ideas could support group-based solving of other "wicked problems" from outside software development.

The KR framework aims to support incremental and iterative solving of large problems for which no one individual possesses sufficient knowledge to reasonably arrive at a workable solution; the explicit connection to software re-architecting is almost accidental. Consider the five dimensions of the framework. The Component dimension does mention "the logical units of implementation" but this could as easily say "the chunks of the solution that are meaningful to the people involved." The Dependency dimension could simply indicate that one chunk uses another chunk in a way that matters to the people involved in the problem solving. "Software" is irrelevant to the other three dimensions.

Whether an arbitrary group problem-solving exercise is amenable to an underlying box-and-arrow diagram approach is an open question, notwithstanding the desires of the purveyors of the Unified Modeling Language (UML); however, note that UML pushes towards precision, which is problematic when information is incomplete, and of questionable value when precise details must be faked. In addition, it is unclear that a box-and-arrow interface will be appropriate for all contexts.

Nevertheless, the essential core of the idea is to force people to write down even their half-baked ideas in a simple but structured manner in order to communicate these to the other people involved, thus supporting feedback/pushback and iteration towards a solution in a fashion that makes sense to them. The explicitness is key: even if someone's idea is not fully-formed, the others can start to understand what is in their mind, permitting constructive arguments, refinements, and alternative ideas. Without the explicitness, confusion and/or chaos can easily reign silently behind the scenes wherein the people all *believe* they understand each other, until that unpleasant moment when they realize that there were inconsistencies in their understanding. Demanding excessive detail or being overly restrictive on the form of the "plans" created is not appropriate; the flexibility and simplicity here seem to work nicely, though it is early days to be too sure of that claim.

VI. Related Work

Much research has been devoted to the topic of *refactoring*: small changes made to narrow aspects of software source code to improve it for developers [11], [12]. Some of these works also refer to another type of internal improvement to programs: *big refactoring* or *large-scale restructuring* that, because of its scale and impacts on a system as a whole, has been raised as a different issue [11], [13], [14]. Unlike the refactoring category where changes are fully identified and specified as a common solution to a common problem in any system implementation, "big refactorings" (i.e., re-architectings) occur for unique reasons and in very different ways in software systems. So, they cannot be categorized and specified beforehand for further support. If this were to be done for a specific kind of re-architecting, its purpose and application must be very limited, limiting its utility.

A part of the software engineering research community has focused on modelling software architecture transformations and evolution [15]–[23]. The purpose of the re-architecting modelling approaches is to describe the transformations as well as the "evolution path" from one type of architecture to another. For this purpose, architecture description languages are expanded or new languages are introduced in order to describe a re-architecting in a formal way and allow researchers to perform a series of verification on the resulting models. The audience of these methods are not developers but architects of mission critical systems whose goals justify rigorous modelling activities.

Another part of the research community has focused on automating parts of the implementation of the changes in a particular programming language [24]–[26]; such works address some specific technical issues. Yet another part of the research community has focused on improving existing support for more general technical issues that are not restricted to software re-architecting, such as better code understanding and visualization, better code analysis, change recommendation, and pragmatic code reuse to name a few [27]–[35]. Still other existing works have focused on the automatic identification of architectural refactoring opportunities within an existing software system; this addresses only a fraction of the triggers suggesting possible support for re-architecting a software system [36]–[38].

Most available empirical studies on this subject have considered some question about *refactoring* [11], [12]. Since re-architecting is a kind of refactoring too, these studies also contribute empirical findings about software re-architecting and the challenges people face in doing it [5], [6], [14], [39]–[41]. These studies commonly indicate that what developers of a re-architecting need is beyond the automation of the code transformations. These studies often suggest that even small-scale refactoring possesses a *human aspect* concerning the troubles of human collaboration in this task and a *technical aspect* which is about the technical problems related to the system or the transformations, justifying our work.

VII. Conclusion

The software architecture of large-scale systems, in particular, can have a direct impact on user experience and key properties like security. When the business context changes, changes to the software architecture can be necessary lest the system become obsolete. Little attention is paid to the human aspects of this problem: who needs to understand the current software architecture, who needs to understand concretely how it needs to become, and how do they determine how to get from the before picture to the after picture considering that it cannot possibly be done by a single person? Evidence from current practice suggests a non-systematic approach that too frequently leads to failure. Instead, support is needed for incremental and incomplete understanding and decision making, dealing with the realities of many people collaborating in a large organization with differing levels of technical- and business-level expertise, sometimes narrowly focused.

We have presented a novel knowledge representation framework, derived from a prior industrial case study and interviews with industrial practitioners, to address the key problems that

recur during software re-architecting—fundamentally human problems only coloured by the technical context, not rooted there. As an initial validation of the KR framework, we conducted a small user study comparing traditional tools for understanding and communicating about software changes, against a paper prototype realization of the KR framework. The results suggest that the KR framework addresses better the problems encountered in software re-architecting, supporting investment in the development and longitudinal study of concrete tools realizing it.

We note that the core ideas of the KR framework are agnostic of the context of software re-architecting; thus, we postulate that the framework could apply to other group problem-solving contexts where collaboration on "wicked problems" is necessary to overcome them. Further study is needed.

Acknowledgments

We thank Kanishka Singh for his assistance with the paper prototype study, and the anonymous reviewers for their helpful comments. The Conjoint Faculties Research Ethics Board of the University of Calgary has approved the human subjects study reported here, as REB 6926. This research was funded in part by a Discovery Grant from NSERC Canada.

References

[1] B. P. Lientz and E. B. Swanson, *Software Maintenance Management: A Study of the Maintenance of Computer Applications Software in 487 Data Processing Organizations*. Addison-Wesley, Reading, MA, 1980.

[2] N. Chapin, J. E. Hale, K. M. Khan, J. F. Ramil, and W.-G. Tan, "Types of software evolution and software maintenance," *J. Softw. Evol. Process*, vol. 13, no. 1, pp. 3–30, 2001.

[3] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

[4] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice (3rd Edition)*. Addison-Wesley Professional, 2012.

[5] M. Leppänen *et al.*, "Decision-making framework for refactoring," in *Proc. IEEE Int. Wkshp. Managing Technical Debt*, 2015, pp. 61–68.

[6] ——, "Refactoring: A shot in the dark?" *IEEE Softw.*, vol. 32, no. 6, pp. 62–70, 2015.

[7] N. A. Ernst *et al.*, "Measure it? manage it? ignore it?: Software practitioners and technical debt," in *Proc. Europ. Softw. Eng. Conf./ACM SIGSOFT Int. Symp. Foundations Softw. Eng.*, 2015, pp. 50–60.

[8] E. Moazzen, "Identifying the problems of software re-architecting and a knowledge representation framework to address them," PhD thesis, University of Calgary, Calgary, Canada, Jun. 2018.

[9] E. Rich and K. Knight, *Artificial Intelligence*. McGraw-Hill, 1991, ISBN: 978-0-07-100894-5.

[10] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[11] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[12] T. Mens and T. Tourwé, "A survey of software refactoring," *IEEE Trans. Softw. Eng.*, vol. 30, no. 2, pp. 126–139, 2004.

[13] M. Lippert and S. Roock, *Refactorings in Large Software Projects: How to Successfully Execute Complex Restructurings*. Wiley, 2006.

[14] M. Kim, T. Zimmermann, and N. Nagappan, "A field study of refactoring challenges and benefits," in *Proc. ACM SIGSOFT Int. Symp. Foundations Softw. Eng.*, 2012, pp. 50–60.

[15] I. Ivkovic and K. Kontogiannis, "A framework for software architecture refactoring using model transformations and semantic annotations," in *Proc. Europ. Conf. Softw. Mainten. Reeng.*, 2006, pp. 135–144.

[16] D. Garlan and B. Schmerl, "Ævol: A tool for defining and planning architecture evolution," in *Proc. ACM/IEEE Int. Conf. Softw. Eng.*, 2009, pp. 591–594.

[17] D. Garlan, J. M. Barnes, B. Schmerl, and O. Celiku, "Evolution styles: Foundations and tool support for software architecture evolution," in *Proc. IEEE/IFIP Working Conf. Softw. Arch./Europ. Conf. Sofw. Arch.*, 2009, pp. 131–140.

[18] T. Mens, J. Magee, and B. Rumpe, "Evolving software architecture descriptions of critical systems," *Computer*, vol. 43, no. 5, pp. 42–48, 2010.

[19] A. McVeigh, J. Kramer, and J. Magee, "Evolve: Tool support for architecture evolution," in *Proc. ACM/IEEE Int. Conf. Softw. Eng.*, 2011, pp. 1040–1042.

[20] J. M. Barnes and D. Garlan, "Challenges in developing a software architecture evolution tool as a plug-in," in *Proc. Int. Wkshp. Developing Tools as Plug-ins*, 2013, pp. 13–18.

[21] J. M. Barnes, A. Pandey, and D. Garlan, "Automated planning for software architecture evolution," in *Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2013, pp. 213–223.

[22] J. M. Barnes, D. Garlan, and B. Schmerl, "Evolution styles: Foundations and models for software architecture evolution," *Softw. Syst. Model.*, vol. 13, no. 2, pp. 649–678, 2014.

[23] J. M. Barnes, "Software architecture evolution," PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2013.

[24] J. Weidl and H. Gall, "Binding object models to source code: An approach to object-oriented re-architecting," in *Proc. IEEE Int. Comp. Softw. Appl. Conf.*, 1998, pp. 26–31.

[25] S. M. A. Shah, J. Dietrich, and C. McCartin, "Making smart moves to untangle programs," in *Proc. Europ. Conf. Softw. Mainten. Reeng.*, 2012, pp. 359–364.

[26] S. M. A. Shah, J. Dietrich, and C. McCartin, "On the automation of dependency-breaking refactorings in Java," in *Proc. IEEE Int. Conf. Softw. Mainten.*, 2013, pp. 160–169.

[27] R. J. Bril and A. Postma, "An architectural connectivity metric and its support for incremental re-architecting of large legacy systems," in *Proc. IEEE Int. Wkshp. Progr. Comprehen.*, 2001, pp. 269–280.

[28] G. C. Murphy, D. Notkin, and K. J. Sullivan, "Software reflexion models: Bridging the gap between design and implementation," *IEEE Trans. Softw. Eng.*, vol. 27, no. 4, pp. 364–380, 2001.

[29] A. Christl, R. Koschke, and M. A. Storey, "Equipping the reflexion method with automated clustering," in *Proc. Working Conf. Reverse Eng.*, 2005, pp. 89–98.

[30] A. De Lucia, F. Fasano, and R. Oliveto, "Traceability management for impact analysis," in *Proc. Frontiers Softw. Mainten.*, 2008, pp. 21–30.

[31] P. Rovegård, L. Angelis, and C. Wohlin, "An Empirical Study on Views of Importance of Change Impact Analysis Issues," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 516–530, 2008.

[32] J. Carriere, R. Kazman, and I. Ozkaya, "A cost–benefit framework for making architectural decisions in a business context," in *Proc. ACM/IEEE Int. Conf. Softw. Eng.*, vol. 2, 2010, pp. 149–157.

[33] R. Holmes and R. J. Walker, "Systematizing pragmatic software reuse," *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 4, 20:1–20:44, 2012.

[34] B. Li, X. Sun, H. Leung, and S. Zhang, "A survey of code-based change impact analysis techniques," *Softw. Test. Verif. Reliab.*, vol. 23, no. 8, pp. 613–646, 2013.

[35] S. Makady and R. J. Walker, "Validating pragmatic reuse tasks by leveraging existing test suites," *Softw. Pract. Exper.*, vol. 43, no. 9, pp. 1039–1070, 2013.

[36] M. Lippert, *Refactoring in Large Software Projects*. John Wiley & Sons, 2006.

[37] R. Mo, Y. Cai, R. Kazman, and L. Xiao, "Hotspot Patterns: The Formal Definition and Automatic Detection of Architecture Smells," in *Proc. IEEE/IFIP Working Conf. Softw. Arch.*, 2015, pp. 51–60.

[38] F. A. Fontana, I. Pigazzini, R. Roveda, and M. Zanoni, "Automatic Detection of Instability Architectural Smells," in *Proc. IEEE Int. Conf. Softw. Mainten. Evol.*, 2016, pp. 433–437.

[39] J. Chen, J. Xiao, Q. Wang, L. J. Osterweil, and M. Li, "Refactoring planning and practice in agile software development: An empirical study," in *Proc. Int. Conf. Softw. Syst. Proc.*, 2014, pp. 55–64.

[40] T. Sharma, G. Suryanarayana, and G. Samarthyam, "Challenges to and Solutions for Refactoring Adoption: An Industrial Perspective," *IEEE Softw.*, vol. 32, no. 6, pp. 44–51, 2015.

[41] J. Chen, J. Xiao, Q. Wang, L. J. Osterweil, and M. Li, "Perspectives on refactoring planning and practice: An empirical study," *Empir. Softw. Eng.*, vol. 21, no. 3, pp. 1397–1436, 2016.

Towards End-User Development for Chronic Disease Management

Daniel Rough
SACHI Research Group
School of Computer Science
University of St Andrews, Scotland
djr53@st-andrews.ac.uk

Aaron Quigley
SACHI Research Group
School of Computer Science
University of St Andrews, Scotland
aquigley@st-andrews.ac.uk

***Abstract*—Although developments in modern medicine continue to reduce premature death from acute illnesses, chronic diseases are now pervading the resultant aging population at a growing rate. Such diseases cannot be cured with drug-based treatment, but can be controlled with patients' regular monitoring of their symptoms and consequent lifestyle changes. However, this level of sustained engagement outside face-to-face appointments places a considerable burden upon patients. Smartphones are suitable platforms to support both patients in engaging with self-management plans, and clinicians in directly monitoring the influence of these plans. Bespoke applications exist for such purposes, yet the diversity in patients' lifestyles and levels of engagement necessitates many new or personalised applications. One approach, to solve these problems at scale, is with end-user development. This paper reports the findings from interviews with clinicians, and ethnographic observation in chronic disease management clinics, to derive requirements of end-user development technology to support clinicians and patients in tailored management of their diseases. Time and quality are key factors towards stakeholders' acceptance of chronic disease management with end-user development.**

I. Introduction

Medical research continues to fight a winning battle against acute deadly diseases. However, the global reduction of *acute* infectious diseases such as smallpox, malaria, measles and polio, has increased overall life expectancy, giving rise to a dramatic increase in the problem of *chronic* diseases. According to the World Health Organization (WHO), global life expectancy increased by 5.5 years from 2000 to 2016[1]. In parallel, almost 60% of worldwide deaths and 43% of the global disease burden are attributed to chronic diseases - figures expected to rise to 73% of deaths and 60% of global disease burden by 2020[2]. With a disparate range of causal factors, various social and economic influences, and no clearly delineated solution, this chronic disease pandemic is, by definition, a "wicked problem".

Technological innovations have transformed the effectiveness of healthcare around the world. Examples such as MRI scanners, laser eye surgery, or bionic hands may spring to mind, but these are undisputed solutions to "tame problems". In this paper, we instead draw attention to technology that can support human-centred communication and collaboration towards developing better approaches to patient management. From an organisational perspective, patient records and prescriptions can be digitised for ease of access and transfer across different practitioners. In terms of diagnostics, wearable devices that quantify and summarise health data such as blood pressure and heart rate can be used to monitor patients more effectively. Smartphone apps are now being developed that can trigger emergency actions based on physiological data, such as automatically calling emergency services when the onset of a heart attack is detected [1]. The expanding opportunities afforded by mobile health technology have even empowered patients to monitor and manage their own conditions, independent of clinician involvement. With increasing hospital waiting times, and the prevalence of chronic diseases in the global population, the UK's National Health Service (NHS) has already begun to act on these potential benefits, through introduction of an NHS-certified app library[3].

While bespoke apps can help chronic disease patients control their symptoms and effectively provide a solution at an individual scale, the diversity in symptoms, lifestyles, socio-economic status, and self-efficacy of these patients in managing their conditions, all preclude a "one-size-fits-all" app solution. Thus, we propose that this problem is ideally suited to end-user development (EUD) technology, where flexible software can be tailored to the requirements of its end-users and adapted dynamically as required. Allowing clinicians to collaborate in the development of disease management apps, personalised to individual patients, would enable a human-centred approach to solving this problem. Indeed, sufficiently motivated patients could themselves engage in EUD activities.

The contribution of this paper is a qualitative analysis of clinician interviews, and observation of nurses running chronic disease management clinics, from which a set of requirements are derived for EUD technology in the management of drug-resistant chronic diseases. In particular, our results address:

- Stakeholders' perceived utility of EUD as a novel form of human problem-solving (*perceived potential*)
- Challenges with respect to patient-clinician communication, as well as inter-clinician communication (*workflow requirements*)

1. www.who.int/gho/mortality_burden_disease/life_tables/situation_trends_text
2. www.who.int/chp/about/integrated_cd Accessed 27/06/18
3. https://digital.nhs.uk/nhs-apps-library Accessed 17/03/18

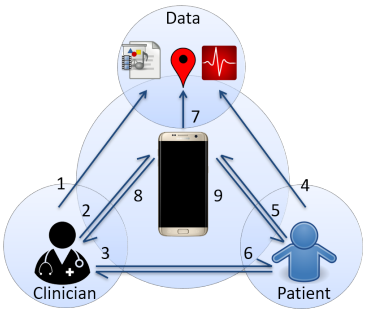

Fig. 1. Various interactions between clinicians, patients, and apps are required

- Organisational factors required to support integration of this technology into current working practices (*facilitating conditions*)

II. Related Work

"The use of mobile and wireless technologies to support the achievement of health objectives (mHealth) has the potential to transform the face of health service delivery across the globe" [2]

This quote from the WHO's report on mHealth innovations exemplifies how app-based solutions to health service issues are being taken seriously by global organisations. Indeed, there are now over 325,000 health-related apps available for download [3], provoking concerns about the lack of evidence-based information in the majority of these apps, and the associated dangers of misinforming their users [4]. In particular, we focus on apps for assisting in non-communicable disease management, which do not directly solve problems in themselves, but instead facilitate "human problem-solving" by encouraging self-reflection, and enabling understanding between patients and clinicians.

A literature review was conducted within computer science literature databases including the ACM, IEEE, and Scopus digital libraries, as well as the PubMed Central and APA PsycINFO databases, using combinations of terms including 'smartphone', 'self-monitoring', 'self-management', 'experience sampling', 'user-centred', 'participatory design' and 'recommendations'. Particular insight was obtained from user-centred design studies that highlighted stakeholder perceptions of smartphone apps for self-management, prompting the addition of the latter three terms.

From this review, a variety of interactions between patients, clinicians, and apps themselves were proposed, as illustrated in Figure 1, representing an evidence-based set of features for general-purpose disease management technology. Arrow numbers correspond to the following numbered descriptions of each interaction.

Clinician Interactions

1) Viewing real-time patient data: Instant access to self-monitored patient data could guide treatment decisions in face-to-face clinical appointments. Study participants living with cystic fibrosis [5] and attention disorders [6] both strongly supported the provision of contextual information to clinicians. In an evaluation study of an app where such information access was implemented, pediatricians described how this saved time, focused appointments, and facilitated communication about difficult issues during these appointments [7].

2) Updating management plan in real-time: Studies that cite the benefits of smartphone-based self-management all describe bespoke apps developed for the specific purpose of each study. Hence it appears that, although these benefits are generic and adaptable, their implementations are not. Thus, the researcher must be able to adapt these features found in bespoke apps to any study they choose to run. In doing so, they perform end-user development (EUD) activities. While there are few studies addressing this interaction, the work of Tetteroo and Markopoulos addresses EUD for rehabilitation therapists to design exercises for their patients [8].

3) Providing feedback to patient: In user-centred design studies of healthcare apps, feedback provision from clinicians to patients was extensively discussed. Given that clinicians have very little time outside of scheduled appointments, there was surprising enthusiasm for this type of interaction. For example, it was recognised that in-the-moment professional assistance on coping with cancer-related pain, prompted by self-assessment data, would support sustained engagement with self-monitoring [9]. Clinicians also suggested that feedback would not have to be a time-critical intervention, and were enthusiastic about providing regular feedback to patients on their data [10]. In an implemented trial, clinicians valued a feature allowing response to alerts on patients' symptoms of potential heart failure, suggesting that the initial workload involved in responding to these alerts would be reduced in the long-term by minimising unnecessary hospital admissions [11].

Patient Interactions

4) Viewing aggregated, collected data: Sophisticated sensing and growing storage capacity of smartphones enable rich visual charts of objective, historical data to be displayed. Visual feedback could raise engagement in patients with chronic health conditions. Participants trialling the *MONARCA* system valued both the ability to correlate their moods with objective data, and to determine the temporal antecedents of low moods [12]. Users with chronic conditions expressed interest in viewing their passively sensed information, particularly in the form of visual graphs, supported by participants with mixed chronic conditions [13] and diabetes [14].

5) Tailoring app to personal preferences: Despite the motivational influence of empowerment to self-manage, maintaining a high level of engagement with apps is still a significant challenge. mHealth apps are easily removed or forgotten about, and patients thus must perceive sufficient value from use if they are to retain engagement. Indeed, recent statistics illustrate that a quarter of apps downloaded are only used once[4]. A balance must be sought between ensuring that app

4. http://info.localytics.com/blog/24-of-users-abandon-an-app-after-one-use

content is based on the input of a professional, while also respecting the preferences of its end-users. User-centred design studies of health apps have elicited the importance of flexible prompt schedules to end-users [15], [16]. Moreover, end-users also desire the ability to control the content of feedback that they are provided with [12], [13], as well as the format in which this feedback is presented [6].

6) Providing feedback to clinician: While visual feedback and personalisation are both useful features for improving engagement, certain issues relevant to patient care may require a direct channel of communication to clinicians. As such, *"healthcare partnership"* was an emergent theme in one focus group study [13]. Focus group participants expressed a desire to contact clinicians for additional explanation of their received feedback. Involving clinicians in this *"sense-making"* process was proposed to support patients in attaining maximum benefit from their management plans. Additionally, individuals with cystic fibrosis identified how self-monitored data would enable them to provide feedback to clinicians between appointments [5].

App Interactions

While supporting human problem-solving by definition relies on ensuring humans can make decisions, actions that can be made automatically by smartphone apps can provide knowledge for human decision-making at key times.

7) Receiving and classifying data: Self-reports can assess intentions, attitudes or certain subjective physical symptoms (such as discomfort or pain). However, an increasing breadth of information can be objectively assessed from sensors built-in to the smartphone, or externally worn. In doing so, passively collected sensor data can minimise self-report burden, provide researchers with richer information, and enable tailored assessment and intervention strategies. For example, Ben-Zeev et al. showed that smartphone-sensed geospatial activity and sleep duration were significantly related to stress levels in a cohort of young people [17]. Adams et al. review smartphone sensing for monitoring physiological and behavioural biomarkers, providing a comprehensive source of examples [18].

8) Sending alerts to clinician: From knowledge acquired through subjective and objective data, disease management apps could inform clinicians of in-the-moment issues or emergency situations. In many conditions, early detection of symptomatic pre-cursors can prevent fatal consequences through clinician intervention. In a user-centred study on app requirements, clinicians and cancer patients both supported a feature to alert clinicians when patients reported high levels of pain [9]. Clinician alerts were also implemented in a randomised control trial, where physiological readings transferred to smartphones via Bluetooth, combined with patients' symptom reports, alerted clinicians to heart failure pre-conditions [11]. The trial reported improved health outcomes, with patients expressing feelings of reassurance and self-efficacy, and clinicians confirming the utility of alerts.

9) Sending notifications to patient: Interventions could be dynamically delivered at locations of interest or on physiological measures. For example, the *Q Sense* app deployed tailored interventions when participants dwelt in identified smoking locations, with post-study feedback exhibiting positive response towards location triggers, and tailored messages [19]. For self-management, users with chronic conditions valued the possibility of contextual reminders, such as those prompted at a particular location [5], [16]. One study had participants describe their physical activity after levels of intense activity, or extended non-activity, were detected with a smartphone accelerometer [20]. From a healthcare perspective, this increased self-awareness could promote positive behaviour [21].

Summary

The cited studies employ user-centred design to solve "tame", albeit non-trivial problems of developing engaging healthcare apps for particular conditions. We suggest that for chronic disease management apps to be effective on a global scale, the extent to which these features are utilised should be determined by clinicians and patients themselves, shifting from design *before* use to design *during* use [22].

However, developing EUD tools that support stakeholders in their current practice, and evaluating the success of these tools in doing so, are major challenges, which require investigation beyond usability studies. In real-world deployments, an individual's interactions with technology are highly dependent on others who form part of their working practices, and existing technologies in this environment. These factors are diverse and dynamic, such that continuous communication and collaboration are necessary to ensure that deployed technology evolves with an environment and the people within it. As a result, the approach taken here was to elicit direct feedback from clinicians, as the potential users of EUD technology.

III. Interviews

Semi-structured interviews were conducted with eight practising clinicians in our university's school of medicine. Interviews took place at the interviewees' location of choice, lasted approximately 45 minutes, and were audio recorded. The recordings were then transcribed, and qualitatively coded for thematic analysis. Coding was structured based on the addressed themes specified in the introduction, which were linked to constructs in validated models of technology acceptance, specifically the Technology Acceptance Model, as shown in Figure 2, and the Unified Theory of Acceptance and Use of Technology [23]. In doing so, factors pertaining to the real-world adoption of chronic disease management EUD were identified. The relevant constructs of these models are *perceived usefulness* (which is further divided into *perceived potential* and *workflow requirements*) and *facilitating conditions*. These themes are defined as follows.

1) **Perceived potential** - Clinicians were given a brief overview of an existing EUD tool, and asked for feedback on the perceived utility of EUD for disease management.
2) **Workflow requirements** - Clinicians were asked about their current practices in dealing with patients managing

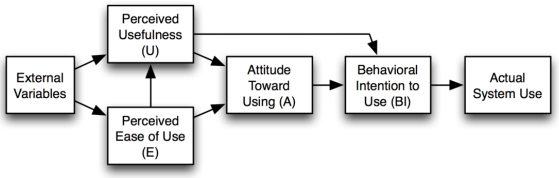

Fig. 2. The Technology Acceptance Model [25]

TABLE I
PERCEIVED BENEFITS/BARRIERS OF EUD TECHNOLOGY FOR CHRONIC DISEASE MANAGEMENT, BY CLINICIANS AND PATIENTS

Stakeholder	Perceived Benefit	Associated Barrier
Patient	Patients more independent	Incapable patients at risk of being forgotten
	Patients empowered to self-monitor	Patients are diverse in self-management needs
Clinician	Clinicians can track patients' progress	Diverse data can overload an individual clinician
	Clinicians acquire more accurate data	Patients may be burdened by extra self-monitoring

chronic conditions, with emphasis on activities that would need to be supported.

3) **Facilitating conditions** - Clinicians discussed the practical issues of integrating novel technology into an existing healthcare system.

Although we demonstrated our specific EUD tool, *Jeeves*, to clinicians in the interviews, the feedback elicited focused on broader issues of its integration into an existing healthcare system. Factors pertaining to existing features of Jeeves are discussed in further detail in our work with psychology researchers [24], and instead the hypothetical features derived in our related work were used as prompts for idea generation in these interviews.

The clinicians interviewed consisted of five general practitioners (GPs), a pharmacist, an ophthalmologist, and a clinical psychologist. A diversity of roles, responsibilities, and opinions with respect to patient care were thus exposed.

A. Perceived potential

Although clinicians and patients are two distinct stakeholders, the potential for improved patient health outcomes would reciprocally benefit the clinicians themselves, in terms of reducing unnecessary appointments and freeing up clinic time. However, opinions regarding the effect of smartphone-based self-management on time were varied, with some clinicians concerned that their workload would be increased. Potential benefits and barriers are summarised in Table I.

C4, a GP, explained the benefits of a *holistic* model of care that takes patients' psychosocial issues and lifestyle habits into account. However, time-constrained appointments, and pressure to see as many patients as possible, means that GPs often rely on quick prescriptions of medication to avoid overrunning on clinic time. A further aggravating issue is that appointment slots are often used for simple reviews:

"I certainly feel like we do epilepsy reviews face-to-face which could be done remotely. Because a lot of that is asking questions or providing advice which actually you don't need to physically be in the same room as somebody to do that"

Thus, when face-to-face diagnoses are not required, as in these epilepsy reviews, self-management technology could increase appointment slots for acutely unwell patients.

Clinicians further explained that patients were often poor at reporting their symptoms and summarising their experience over a number of weeks:

"the doctor sees them every two weeks so then some patients they tell them 'how was everything during the two weeks?','Yes everything was alright'...because then time passes and patients tend to forget" (C2)

Irrespective of patients' memory bias, the act of taking an individual reading in a clinical setting is insufficient to characterise the overall status of the patient. C8 explained how, for diabetic patients, a single "HBA1C" reading is relied upon to derive longitudinal information:

"we're relying on one single value of this HBA1C so...you might know that the patient has not been well-controlled but you don't know why..."

B. Workflow requirements

Although the perceived potential of EUD technology was evident from clinicians' feedback, a number of requirements were addressed towards the workflows and communication that would need to be supported. Such requirements, and barriers in Table I that they would address, are shown in Table II.

1) ***Technologies should make it easy for clinical staff to communicate and collaborate with each other****:* Chronic disease patient care is a collaborative effort between GPs and specialist practice nurses. Clinicians explained how individual diseases were dealt with by nurses who had received specific training for management of that disease. Patients may also see different doctors, so it is important to enable collaboration and understanding between clinicians. Current technology does not provide a simple means of doing so, as expressed by C3, a clinical psychologist:

"if we make a change to patients' medication, the quickest way to communicate that to the GP is to fax them, would you believe. So, if we email the GP, we have no idea when the GP might pick that up" (C3)

It also emerged that comorbidities would need to be taken into account. Patients with a chronic disease such as diabetes often have a coexisting disease such as asthma or hypertension. While one nurse would be capable of authoring an action plan for managing a patient's diabetes, this would require input from an asthma nurse if such a comorbidity was present.

2) ***Technologies should support patients' current self-monitoring****:* Clinicians also expressed positive opinions with respect to the feasibility of self-monitoring, given that many patients are already engaged in doing so. It emerged that

TABLE II
EUD WORKFLOW REQUIREMENTS FOR INTERVIEWED CLINICIANS

EUD Support	Barrier Addressed	Functionality Required
Clinician collaboration	Patients are managed by many clinicians who need specific types of data	Community support, including shared editing + collaboration functions
Monitoring Integration	Patients may be burdened by additional work on top of current self-monitoring	Simple integration with existing monitoring equipment and routines
Individual Tailoring	Patients are diverse in their personal management needs	Protocols that are tailored to stages of patient independence
Personalised Reminders	Dependent, incapable patients are at risk of being forgotten	Prompts delivered at times personalised to patients' preferences

patients with chronic diseases are often highly skilled in their management, and willingly purchase monitors to facilitate their independence. In order for new technologies to become an asset to their healthcare, rather than a burden, it should be possible to integrate technologies with patients' current self-monitoring practices. It was suggested that Bluetooth and other wireless data transfer technology could enable patients' results to be seamlessly uploaded from their existing devices.

3) ***Technologies should allow tailoring to individual patients****:* A patient-centred care approach relies on the incorporation of patients' own goals, motivations and constraints, which determine whether or not they are likely to engage in treatment. C6 described these factors as the patient's "agenda":

"Healthcare people, not just doctors, have agendas, but the patients often come in with a totally different agenda, so it's about sorting out what the patient's agenda is and how you can use that to improve their physical wellbeing."

4) ***Technologies should support personalised reminders****:* For the inevitable cohort of patients who would not engage in self-monitoring, it should be simple to set up reminders for different aspects of their healthcare. C3 mentioned that either patients themselves, or their carers, would add appointment reminders on their mobile devices.

"Interestingly quite a few of our patients say they've set reminders on their phone for things like daily reminders of it's time to take their medication. So they've actually done that themselves, or somebody's usually done it for them..."

Additionally, many of these patients are required to take medication at particular times in their daily routine. C8 suggested that it should be possible for patients to input their waking and sleeping times, as well as regular mealtimes, to ensure that medication is taken.

C. Real-world integration

In the context of clinical practice, constraints imposed by the NHS present significant barriers to EUD adoption in the UK. Healthcare transformation is difficult in a system where it must take place at a national level. Thus, facilitating conditions must be fulfilled to enable the transition from clinicians' *intention to use* to their active *usage behaviour* [23].

1) ***Clinicians must be allowed time to get used to new technologies****:* Although time was a critical factor across all clinicians, GPs in particular reported a lack of time given their high workload. Ironically, clinicians would struggle to adopt time-saving technology because the initial time to introduce it would be too costly, as expressed by C4:

"...I know that takes like a minute or two, it's a minute or two I might not have. The times are very very precious...this will free up time ultimately. But it's that initial jump, isn't it?"

GPs expressed frustration that, paradoxically, they need more time to discuss general health management that would reduce unnecessary appointments. However, this high number of appointments forces them to limit individual patient time to 10 minutes.

2) ***Clinicians must trust the resilience of apps developed with EUD****:* From a technical perspective, clinicians were wary about the reliance on technology to capture abnormalities and give patients immediate feedback at critical times. Quality assurance of smartphone apps is critical in a medical context, where the health and well-being of patients could potentially be put at risk. Clinicians had concerns about what would happen if such an app were to malfunction:

"if suddenly you don't have network coverage, then your reading might not get to the platform or whatever or the server in two hours, and then if it's a high reading...you might not be able to receive the message soon enough" (C8)

In summary, the clinician interviews gave further insight into the types of interactions that would be required to support effective human problem-solving in chronic disease management. In order to strengthen the ecological validity of these findings, it was necessary to observe the current practices of clinicians involved in this domain.

IV. OBSERVATION

An observation session was conducted over two hours at a local clinical practice, with one hour assigned to both a nurse running a diabetes clinic, and a nurse running a hypertension clinic. Within each hour, three patients were seen by both nurses. Audio recordings and computer use were not permitted within the clinical setting. Instead, notes were hand-written during the observation sessions, transcribed and expanded on within 24 hours of the recording process. The observation sessions were *naturalistic*, in that the observation aimed to disrupt the process as little as possible.

A. Diabetes management clinic

The observations made during the diabetes management clinic are divided into the following themes: self-management, salient problems, and technology use.

1) Self-management: The nurse explained that, contrary to the stereotype of older patients being less accepting of patient-centred care, diabetic patients of all ages are proactive in self-monitoring their conditions, and often enthusiastic in doing so. Patients are required to carry electronic glucose monitors with them, in order to ensure that their levels are safe prior to driving or engaging in other activities that would require

TABLE III
CLINIC OBSERVATIONS AND THEIR RELATION TO WORKFLOW REQUIREMENTS

Observation	Patients are already self-monitoring, but have to bring their readings on paper
Implication	Allowing patients to synchronise readings on their various electronic monitors would improve the process for patients and reduce errors
Requirement	**Technologies should support patients' current self-monitoring**
Observation	Patients' medication details and recent results are not always synchronised
Implication	New technology in clinical practice would ideally allow clinicians to immediately view updates made by another clinician
Requirement	**Technologies should make it easy for clinical staff to communicate and collaborate with each other**
Observation	Patients are burdened with keeping track of many responsibilities
Implication	A function to send time-based or context-sensitive reminders could improve appointment and medication adherence
Requirement	**Technologies should support personalised reminders**
Observation	Failure to remember medication or monitoring equipment could be life-threatening
Implication	If clinicians develop reminder prompts for patients, it is vital that these arrive consistently if patients' health is at stake
Requirement	**Clinicians must trust the resilience of apps developed with EUD technologies**
Observation	Clinical nurses have very little time in between appointments
Implication	If nurses are going to employ EUD to save future time, they must be allocated scheduled time to monitor patients and track compliance
Requirement	**Clinicians must be allowed time to get used to new technologies**
Observation	Patients have a variety of comorbidities, requirements and personal schedules
Implication	A generic management app may not be appropriate for diverse requirements of comorbid patients with individual demands
Requirement	**Technologies should allow tailoring by individual patients**

concentration, which these patients accept as part of their management regime. As previously mentioned by clinicians, patients often purchase their own monitors for home use.

2) Problems: Contrary to the expectation of non-compliance with self-monitoring, patients are highly compliant, but face barriers in terms of memory and information overload. Two of the three patients had their medication reviewed, revealing a considerable number of drugs for managing comorbidities present in these patients. The nurse required clarification from patients on medication they were currently taking, as medication lists are sometimes not synchronised across the practice. When these discrepancies are not checked, this can result in patients reordering unnecessary prescriptions.

3) Technology: In terms of resources, patients were provided with a self-management plan on paper. The nurse explained that while patients were generally enthusiastic about a personalised plan, the paper often got misplaced, so that the nurse would keep a copy herself to use for discussion with patients. Indeed, technology use was limited to the nurse's patient management application. She described how some clinics send patients SMS appointment reminders, but that this is at the discretion of the clinic's management, and due to the logistics of implementation, is uncommon.

B. Hypertension management clinic

The hypertension management clinic had a similar format to that of the diabetes clinic, involving a nurse with expertise in hypertension, who saw three patients for their annual reviews.

1) Self-management: Despite similar difficulties to the diabetic patients in independent management of their treatment, hypertension patients also appeared to be proactive in doing so. All three patients were satisfied with monitoring their blood pressure readings every day, expressed interest in the results, and engaged in active discussion with the nurse, rather than passively receiving information. The nurse explained that patients frequently get *white coat hypertension* - the artificial raising of blood pressure readings caused by the unfamiliar clinical environment. To alleviate this, they are encouraged to take their blood pressure readings at home, in a familiar context that provides the most accurate results.

2) Problems: Forgetfulness, particularly in medication management, was a pertinent issue noted during observation in two older patients. As discussed in the diabetes clinic, patients would order unnecessary medication, due to lack of communication between clinical staff. One patient was unsure of the medication she was regularly taking. The nurse was required to go through this patient's list of medication, asking which were actually necessary. Although appointment reminders were sometimes made via phone calls, this placed additional responsibility on nurses beyond their standard schedule.

3) Technology use: The use of technology again appeared to be limited to the nurse's patient management application, which did not integrate well with patients' paper-based readings, as previously described in the manual transcription, scanning and uploading of patient results. The nurse described a new system for prescription management, whereby patients receive a text message to remind them that their prescription is due. The system has so far received a positive response from patients, but is limited to prescriptions and does not take into account regularly scheduled appointments, or reminders to take medication, for example.

C. Summary

These observations are not an in-depth ethnographic study but instead served as a source of triangulation with clinician interviews. Table III summarises observations, their implications, and the requirements derived from interviews that they support, described in detail as follows.

The between-patient variation in skills and requirements necessitates individually tailored apps. However, *within*-patient differences of knowledge and proactive behaviour over time indicate the utility of allowing patients to tailor their apps independently. Recently diagnosed patients require considerable education and reminders to engage in treatment, but over time, their self-efficacy increases such that monitoring and medication become routine activities. Thus, continuous reminders from an app at this stage are likely to be irritating and intrusive. Allowing patients to adjust the level and type of reminders as necessary would therefore be beneficial, represented by the requirement that ***"technologies should allow tailoring by individual patients"***.

The laborious transfer from paper to electronic health records could be alleviated for nurses, but if patients have to do their own transfer of readings from glucose or blood pressure monitors into new technologies, there still exists labour on their part. Further, the requirement that ***"technologies should support patients' current self-monitoring"*** could also streamline this process for patients themselves.

This difficulty in information transfer was also present between nurses and other clinicians. For example, changes to medication initiated by GPs were not immediately communicated through the clinic's system, such that nurses relied on patients to provide up-to-date information. Accessing results such as blood tests also appeared to be a time-consuming process for nurses. This difficulty in communication was highlighted through interviews, motivating the requirement that ***"technologies should make it easy for clinical staff to communicate and collaborate with each other"***.

Allowing patients and nurses to work together to schedule reminders on patients' devices for medication, appointments, and emergency equipment could be a useful feature, consistent with the requirement that ***"technologies should support personalised reminders"***. As an example, a prompt could be issued to a patient when they leave their home to remind them to take appropriate equipment with them. This further links with the importance of app reliability. Ensuring that reminders are sent to patients consistently is of particular importance, such that ***"clinicians must trust the resilience of apps developed with EUD technology"***.

The adoption of problem-solving technology must not impose additional burden on clinicians in its use. Although empowering patients to take responsibility for their healthcare would reduce the burden of unnecessary or missed appointments, it appears that the primary barrier to the adoption of novel technology is the initial burden of introduction. Nurses are still constrained by time, and as such are unlikely to monitor patients' self-reported readings in between appointments unless specific time was allocated for them to do so. This supports the requirement that ***"clinicians must be allowed time to get used to new technologies"***.

V. Discussion

For clinicians to adopt new problem-solving technologies in practice, our interviews suggest that this is largely dependent on the *facilitating conditions* determined by their organisation. Clinicians are constrained by the health service's stringent requirements on app evaluation and ethical considerations. Nevertheless, the WHO's advocacy of mobile technology's role in healthcare and the establishment of the NHS app library, suggest that technologies that mitigate the chronic disease pandemic will be feasible in the near future.

Clinicians explained that target patients are often already active in self-monitoring, suggesting that, although such technology would be feasible, clinicians also have initial requirements that must be satisfied by software to justify the disruption of their current working practices. We briefly discuss these requirements in relation to the two overarching themes of *time* and *quality* of EUD technologies.

Time appears to be the most critical barrier faced by clinicians, thus the time EUD would ultimately save (*perceived usefulness*) and the time that organisations would allow for integration (*facilitating conditions*) are determining factors for adoption. The NHS, or indeed any organisation that a clinician is part of, must allocate time for a period of adjustment. However, an organisation's willingness to do so is contingent on the assurance that it will eventually save time, and therefore money. Empirical evidence of time-saving capabilities through an NHS-supported evaluation is thus necessary. Although clinicians will still have limited time to engage in development activities, patients' motivation to independently manage their health suggests that these patients could themselves act as developers, tailoring their management applications to their own goals and needs.

Quality is another overarching factor discussed. The quality of an app in terms of its functionality is a determining adoption factor (*perceived usefulness*), but particularly in terms of its reliability. A reliable app ensures that constant debugging and patient frustration are minimised, but is also necessary to ensure that apps will not potentially cause harm by malfunctioning (*facilitating conditions*). Reliability is another critical facilitating condition for organisations. Particularly in the health service, all apps must undergo rigorous evaluation to ensure that they will do no harm to patients. This is a difficult implication to address for an EUD tool, given that adoption is contingent on not just the reliability of one particular app, but on the reliability of all apps that could be developed. Evidence that clinicians and patients cannot implement harmful apps is therefore a necessary adoption factor.

A. Limitations

Although interviews and observation offer insights beyond that of usability studies, some limitations remain. Half of the interviewees were GPs, which was a consequence of convenience sampling in our university's school of medicine. It emerged that GPs are minimally involved in chronic disease management of patients. Nevertheless, all clinicians were well-informed on the duties of practice nurses and therefore aware of their potential motivations and constraints. Further work is required to elicit the firsthand feedback of practice nurses.

Furthermore, unlike the studies in our Related Work section, we did not receive direct feedback from individuals currently managing chronic diseases. However, clinicians provided detailed accounts of potential issues and benefits for patients, which were reflected in the observation of such patients, and were adequate for addressing adoption factors for clinicians.

Finally, as chronic disease is a global problem, we are limited in the scope of our research within the UK health service. Attitudes towards, and understanding of health technologies will vary widely across different cultures, constituting an extra dimension of 'wickedness'. Replicating this work with international clinicians would provide insight into the breadth of this variability.

VI. Conclusion

EUD platforms can support clinicians and patients to take charge of chronic disease management at a large and complex scale, where there is no clearly delineated and agreed solution for all problem owners. In this paper, we described how clinicians and patients could employ EUD to improve chronic disease management at an individual granularity, with requirements towards real-world adoption. While significant work is required to enable a real-world evaluation, our research thus far suggests a key area of global impact for EUD technologies.

References

[1] M. Gusev, A. Stojmenski, and I. Chorbev, "Challenges for development of an ecg m-health solution," *Journal of Emerging Research and Solutions in ICT*, vol. 1, no. 2, pp. 25–38, 2016.

[2] World Health Organization, "mHealth: New horizons for health through mobile technologies," *Observatory*, vol. 3, pp. 66–71, 2011.

[3] Research2Guidance, "mHealth economics 2017 - current status and future trends in mobile health," 2017, accessed August 20th 2018. [Online]. Available: https://research2guidance.com/product-category/mhealth/digital-health-market-overview/

[4] M. N. Kamel Boulos, A. C. Brewer, C. Karimkhani, D. B. Buller, and R. P. Dellavalle, "Mobile medical and health apps: state of the art, concerns, regulatory control and certification," *Online Journal of Public Health Informatics*, vol. 5, no. 3, p. e229, 2014.

[5] M. E. Hilliard, A. Hahn, A. K. Ridge, M. N. Eakin, and K. A. Riekert, "User preferences and design recommendations for an mHealth app to promote cystic fibrosis self-management," *Journal of Medical Internet Research*, vol. 16, no. 10, p. e44, Oct 2014.

[6] L. Simons, A. Z. Valentine, C. J. Falconer, M. Groom, D. Daley, M. P. Craven, Z. Young, C. Hall, and C. Hollis, "Developing mHealth remote monitoring technology for attention deficit hyperactivity disorder: a qualitative study eliciting user priorities and needs," *JMIR mHealth and uHealth*, vol. 4, no. 1, p. e31, Mar 2016.

[7] S. C. Reid, S. D. Kauer, A. S. Khor, S. J. C. Hearps, L. A. Sanci, A. D. Kennedy, and G. C. Patton, "Using a mobile phone application in youth mental health: An evaluation study," *Australian Family Physician*, vol. 41, no. 9, pp. 711–714, Dec 2012.

[8] D. Tetteroo, P. Vreugdenhil, I. Grisel, M. Michielsen, E. Kuppens, D. Vanmulken, and P. Markopoulos, "Lessons learnt from deploying an end-user development platform for physical rehabilitation," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 4133–4142.

[9] L. A. Jibb, B. J. Stevens, P. C. Nathan, E. Seto, J. A. Cafazzo, and J. N. Stinson, "A smartphone-based pain management app for adolescents with cancer: Establishing system requirements and a pain care algorithm based on literature review, interviews, and consensus," *Journal of Medical Internet Research*, vol. 16, no. 3, p. e15, Mar 2014.

[10] D. Swendeman, S. Farmer, D. Mindry, S.-J. Lee, and M. Medich, "HIV Care Providers' Attitudes regarding Mobile Phone Applications and Web-Based Dashboards to support Patient Self-Management and Care Coordination: Results from a Qualitative Feasibility Study." *Journal of HIV and AIDS*, vol. 2, no. 4, pp. 1310–1318, Oct 2016.

[11] E. Seto, K. J. Leonard, J. A. Cafazzo, J. Barnsley, C. Masino, and H. J. Ross, "Developing healthcare rule-based expert systems: Case study of a heart failure telemonitoring system," *International Journal of Medical Informatics*, vol. 81, no. 8, pp. 556–565, Aug 2012.

[12] J. E. Bardram, M. Frost, K. Szántó, M. Faurholt-Jepsen, M. Vinberg, and L. V. Kessing, "Designing mobile health technology for bipolar disorder: a field trial of the monarca system," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Paris, France, 2013, pp. 2627–2636.

[13] S. W. Miyamoto, S. Henderson, H. M. Young, A. Pande, and J. J. Han, "Tracking Health Data Is Not Enough: A Qualitative Exploration of the Role of Healthcare Partnerships and mHealth Technology to Promote Physical Activity and to Sustain Behavior Change," *JMIR mHealth and uHealth*, vol. 4, no. 1, p. e5, Jan 2016.

[14] D. K. King, D. J. Toobert, J. D. Portz, L. A. Strycker, A. Doty, C. Martin, J. M. Boggs, A. J. Faber, C. R. Geno, and R. E. Glasgow, "What patients want: relevant health information technology for diabetes self-management," *Health and Technology*, vol. 2, no. 3, Sep 2012.

[15] L. Dennison, L. Morrison, G. Conway, and L. Yardley, "Opportunities and challenges for smartphone applications in supporting health behavior change: qualitative study." *Journal of Medical Internet Research*, vol. 15, no. 4, p. e86, Apr 2013.

[16] N. Ramanathan, D. Swendeman, W. S. Comulada, D. Estrin, and M. J. Rotheram-Borus, "Identifying preferences for mobile health applications for self-monitoring and self-management: focus group findings from HIV-positive persons and young mothers," *International journal of medical informatics*, vol. 82, no. 4, pp. e38–e46, 2013.

[17] D. Ben-Zeev, C. J. Brenner, M. Begale, J. Duffecy, D. C. Mohr, and K. T. Mueser, "Feasibility, acceptability, and preliminary efficacy of a smartphone intervention for schizophrenia," *Schizophrenia Bulletin*, vol. 40, no. 6, pp. 1244–1253, Nov 2014.

[18] Z. W. Adams, E. A. McClure, K. M. Gray, C. K. Danielson, F. A. Treiber, and K. J. Ruggiero, "Mobile devices for the remote acquisition of physiological and behavioral biomarkers in psychiatric clinical research." *Journal of Psychiatric Research*, vol. 85, pp. 1–14, 2017.

[19] F. Naughton, S. Hopewell, N. Lathia, R. Schalbroeck, C. Brown, C. Mascolo, A. McEwen, and S. Sutton, "A context-sensing mobile phone app (q sense) for smoking cessation: A mixed-methods study," *JMIR Mhealth Uhealth*, vol. 4, no. 3, p. e106, Sep 2016.

[20] G. F. Dunton, E. Dzubur, K. Kawabata, B. Yanez, B. Bo, and S. Intille, "Development of a smartphone application to measure physical activity using sensor-assisted self-report," *Frontiers in Public Health*, vol. 2, p. 12, 2014.

[21] P. Klasnja and W. Pratt, "Healthcare in the pocket: mapping the space of mobile-phone health interventions," *Journal of Biomedical Informatics*, vol. 45, no. 1, pp. 184–198, 2012.

[22] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, "End-user development: An emerging paradigm," in *End User Development*, H. Lieberman, F. Paternò, and V. Wulf, Eds. Dordrecht: Springer Netherlands, 2006, pp. 1–8.

[23] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, "User acceptance of information technology: Toward a unified view," *MIS Quarterly*, vol. 27, no. 3, pp. 425–478, Sep. 2003.

[24] D. Rough and A. Quigley, "End-user development in social psychology research: Factors for adoption," in *Visual Languages and Human-Centric Computing (VL/HCC), 2018 IEEE Symposium on*. IEEE, 2018, (in press).

[25] V. Venkatesh and F. D. Davis, "A theoretical extension of the technology acceptance model: four longitudinal field studies," *Management science*, vol. 46, no. 2, pp. 186–204, 2000.

Collaborative Problem-Solving Technologies: A Taxonomy of Issues

Steven L. Tanimoto
University of Washington
Seattle, Washington 98195
Email: tanimoto@cs.washington.edu

Sandra B. Fan*
University of Washington
Seattle, Washington 98195
Email: sbfan@cs.washington.edu

***Abstract*—This paper defines several types of systems and technologies that support humans in solving complex problems, and then presents a set of issues relevant to the design of these systems. The issues include how to structure workflows, crowdsourcing, autonomous agents, education and training, version control, collaboration and problem formulation processes. The aim of the paper is to facilitate discussion and future research in the design of effective systems to help humans solve difficult problems, particularly those involving global challenges such as climate change, world poverty, nuclear weapons proliferation, and fake news.**

I. Introduction

The purpose of this paper is to identify and describe a collection of issues that designers and engineers need to address in order to create effective new systems to support humans in solving challenging problems. We hope this paper will facilitate constructive discussions and ultimately lead to improved designs for powerful collaborative problem-solving tools.

Before we define various types of systems and technologies for problem solving, let us describe some of the global challenges that such systems should be able to address, and clarify the context in which to situate this discussion.

A. Global Challenges

How can we use use technology to facilitate solving global problems such as pollution of the oceans, nuclear proliferation, climate change, fake news, and drug-resistant bacteria? Solving these problems is difficult not only because they involve scientific modeling and understanding, but because there are opposing stakeholders that want different or no solutions to the problems. For example, certain stakeholders may be able to push their viewpoints by generating fake news content, and others are able to profit financially from it. At the same time, proliferation of such content may have negative effects on society. New technologies, systems, and processes may be able to help in resolving conflicts and gaining understanding of problems and possible solutions. This paper addresses the question of what sorts of possibilities and challenges there are in designing new systems to help people solve complex problems.

This paper represents a part of the proceedings of the Workshop on Designing Technologies to Support Human Problem Solving: DTSHPS'18.

*Dr. Sandra Fan at the time of this publication is with Google Seattle.

B. Supporting a Culture of Problem Solving

Although an important end goal is to solve specific problems such as arresting global warming, a more accessible and broad goal is to help foster increased awareness and skill on the part of humans to engage productively in problem solving activities. Insofar as stakeholder conflicts are a major roadblock to solving some problems, a change in human understanding and attitudes is an important step towards solving such problems. Even when many conflicts cannot be quickly resolved, the identification of shared goals or shared subgoals can provide milestones along the road to possible broader agreements.

C. Convergence of Technologies

As many computing technologies continue to advance, such as artificial intelligence, supercomputers, software development tools, and interaction devices, there is an opportunity to do more to support problem solving than has been possible in the past. However, it is not clear how best to integrate these technologies to help people solve complex problems. It is timely to address the question of how best to bring together new technologies so as to take advantage of them for problem solving.

II. Definitions

Here we provide working definitions of terms that we use later in the paper. We will ground these definitions using fake news as an example problem.

A. Problem

A *problem* is a need, identified or not. It could be the need to obtain money or some physical object or to obtain the change in some aspect of the state of the world or the state of a puzzle or virtual world.

In terms of fake news, the problem is the need for people who consume news to be aware of who generated the content and what their motives may be.

B. Problem formulation

A *problem formulation* is a description or representation of a problem, typically in a way that makes the problem "actionable" or that permits recognition of possible solutions. Formulations may be informal (e.g., textual descriptions) or formal (e.g., executable code).

There are many ways to formulate the problem of fake news as it is a wicked problem (defined in II-D). We could informally say that our problem formulation is to find ways to show news consumers the provenance of new articles, or to educate people on how to distinguish fake news, or that we need to socially or legally disincentivize fake news creation or propagation. One formal way to formulate the problem might be, we could say that we would like to build a newsreader that can filter out unreliable news sources. Possible solutions would correspond to different features that may be implemented in the newsreader.

C. Problem space

A *problem space* is a set of possible "states" or configurations of problem elements that includes the starting situation and potentially includes one or more goal states. A problem space may be finite or infinite, depending on the nature of the formulation.

If we continue the idea of constructing a reliable newsreader, our problem space would consist of the possible combinations of potential features in the system. Some examples of these features include: automated (or manual) validation of a user's identity before allowing them to post content, only allowing content from a whitelist of certain sources, asking gatekeeper editors to fact-check content, crowdsourcing fact-checking of articles, natural language processing or machine learning of content for reliability, and so on.

D. Wicked problem

A *wicked problem* is a problem that meets certain criteria [2] that make it particularly difficult to solve. We will define them further in IV-C. Common characteristics of wicked problems are: difficulty in formulating the problem, lack of a clearly right or wrong solution, only better or worse ones, and conflict where opposing stakeholders impede reaching consensus on solutions.

Fake news [5] is a wicked problem in that the issue could be formulated in different ways as described in II-B. There is also no single correct solution. Sometimes it may not be possible to fully verify a news source; does that mean it should be hidden? Some beliefs clearly known to be true are later disputed by further scientific research or additional information. All content creators have their own biases; does this mean it is best to limit news to only concrete facts–dates and times of events–and no analysis of the implications of these events? These questions can perhaps only be answered on a spectrum and in combination with each other.

E. Stakeholders

A *stakeholder* is a party (person or group) to a process that has the power to act on solving a problem, and which has an interest (e.g., financial, healthwise, or ideological) in the outcome of the process. With regards to the problem of fake news, there are content consumers who read the articles, each with their own motivation for doing so–gathering information to make a voting decision, or trying to understand how a policy or news event might affect them. Another stakeholder is the content creator, who may be trying to make a living in some way off of their content, or convince others of a viewpoint. The people who are the subjects of news content–public figures, or sometimes everyday people–are also stakeholders. Their stories are the ones being told to others, and this can affect their lives. All of these stakeholders have different interests in the content.

F. AI Service, AI Solving Agent, ML Agent

An *AI service* is a computational function, typically involving inference, or pattern classification, but which may involve solving a pre-formulated problem. An AI *solving agent* is a computational process that not only can perform one or more AI services, but which may take initiative by watching for opportunities to act and then taking actions at those times.

An *ML agent* is an AI solving agent which performs machine learning. This means that it performs data mining, clustering, classifier training, probability estimation, or rule induction. In our example of a newsreader framework, we might design NLP or ML algorithms for detecting the reliability of a news article.

G. Technical Transparency

An AI service, AI solving agent, or ML agent is *technically transparent* when the function(s) it computes are easily identifiable to users, rather than hidden from them, either intentionally or simply due to the nature of the function. Even if the definition or implementation of such functions are available for end users to examine, it may be difficult to understand their implications.

H. Stakeholder Transparency

A problem formulation, problem solution, AI service, AI solving agent, ML agent or data set possesses *stakeholder transparency* when its biases in favor of or against particular stakeholders are readily apparent, typically because of explicit disclaimers. An ML agent may be biased by its training data.

III. System Types

A. Brokering Systems

Systems that help connect problems with people who can solve them are problem "brokers." For example, Innocentive.com, in operation since 2001, runs a brokering service in which users can post problems they would like help with solving. Most of the problems are engineering or chemistry oriented, but others are business-process oriented. Solvers are users who compete with other solvers to win a prize for a best solution.

B. Automatic Solvers

An automatic solver is an AI service or solving agent that takes a suitably formulated problem and tries to return either a legal goal state or a lowest-cost path to a goal state for the problem. Automatic solvers typically use technologies such as heuristic search, simulated annealing, genetic algorithms, case-based reasoning, and/or constraint satisfaction.

C. Collaborative Solving Managers

A system that administers the efforts of a group of humans and/or AI solving agents in exploring a problem space is called a *collaborative solving manager*. An example is CoSolve [3], a web interface for collaboratively exploring a visually-depicted problem space. Users could formally formulate problems (III-D), and then traverse a graph, generating nodes that represent solutions or steps toward a solution. Affordances for collaboration included the ability to see, comment on, and rate these nodes, or to take turns generating the steps toward a solution.

D. Posing Tools

Problem formulation is often the most difficult part of solving a problem. Tools and methodologies supporting formulation we call *posing tools*. Methodologies include ones used in mathematics and science (e.g., see Jonassen). CoSolve's problem formulation interface was in the form of generated Python code snippets that the problem posers could edit to try out different ways of representing their problem.

E. Online Community Support

Mathematician Tim Gowers started the Polymath project to engage the mathematics research community in solving several open problems [6]. An important communication tool in this effort was Gowers' blog, in which he summarized the progress on solving, so that the community efforts could be coordinated.

In addition to blogs, communication methods such as email, discussion forums, and chat rooms can be used as means for sharing status and intentions on the part of solvers in a joint effort. When these communication tools can be effectively integrated with solving activities, some of the difficulties related to identifying session objects, locations in documents, versions and contexts can be avoided.

Koios.org [8] hosts a forum organized according to various social problems that have been identified by users. The site encourages problem solving by identifying top solvers and posting their names.

F. Crowdsourcing Management

Certain problems such as galaxy surveys are well-suited to the citizen science collaboration structure used by Galaxy Zoo. Here, the overall survey problem divides up nicely into thousands of microproblems each of which can be solved independently. The system keeps track of the problems and employs redundancy in the crowdsourcing to control the reliability of results.

G. Data Management

As more and more problems involve digital data, the management of such data becomes important in solving the problems. While it is out of the scope of this paper to characterize the wide variety of such systems, the role that these systems can play in problem solving must be accounted for here.

H. Education and Training

Due to the complexity of global-challenge problems, and due to the need to have humans involved in solving these problems, a major consideration in the design of technologies for problem solving is to help the human users of the system to understand all of the following: the key aspects of the problem being solved, the processes being followed (both by the people involved and the computational parts of the system), and the viewpoints and interests of all stakeholders to the problem. Learning about the problem comes both from external materials that are problem-specific, and exercising the system using existing formulations of the problem to gain an understanding of some of the problem's possible problem spaces. Learning about the processes embedded in the system can come both from tutorials about the system and practice with the system. Finally, understanding the stakeholders involved can come from reading stakeholder descriptions hosted by the system, or reading external materials. The system might host interactive conversations, or prepared videos and other media that help explain other stakeholders' points. Provisions for stakeholder transparency can also contribute to this key aspect of human understanding within a problem-solving environment.

I. Version Control

Both problem formulations and solution activities are subject to frequent revision and negotiation, especially when the problems being addressed are complex. The ability to manage multiple variations of these objects is essential. Version control tools are commonly used in software engineering environments. Similar facilities are needed in complex problem solving.

IV. Taxonomy of Issues

This section revisits many of the topics already discussed, but it lists them in a somewhat more systematic order, with a shallow hierarchy. This may make it easier to point to particular design issues and see them in a context of other issues relevant to the design of collaborative problem-solving systems. Fig. 1 shows the hierarchy as a tree.

A. Problem-Space Theory

1) Identifying and developing theory components most relevant to DTSHPS: Some artificial intelligence (AI) theory is more relevant to automated problem solving than human problem solving, but where humans interact with AI, the theory may be important in facilitating the interaction.

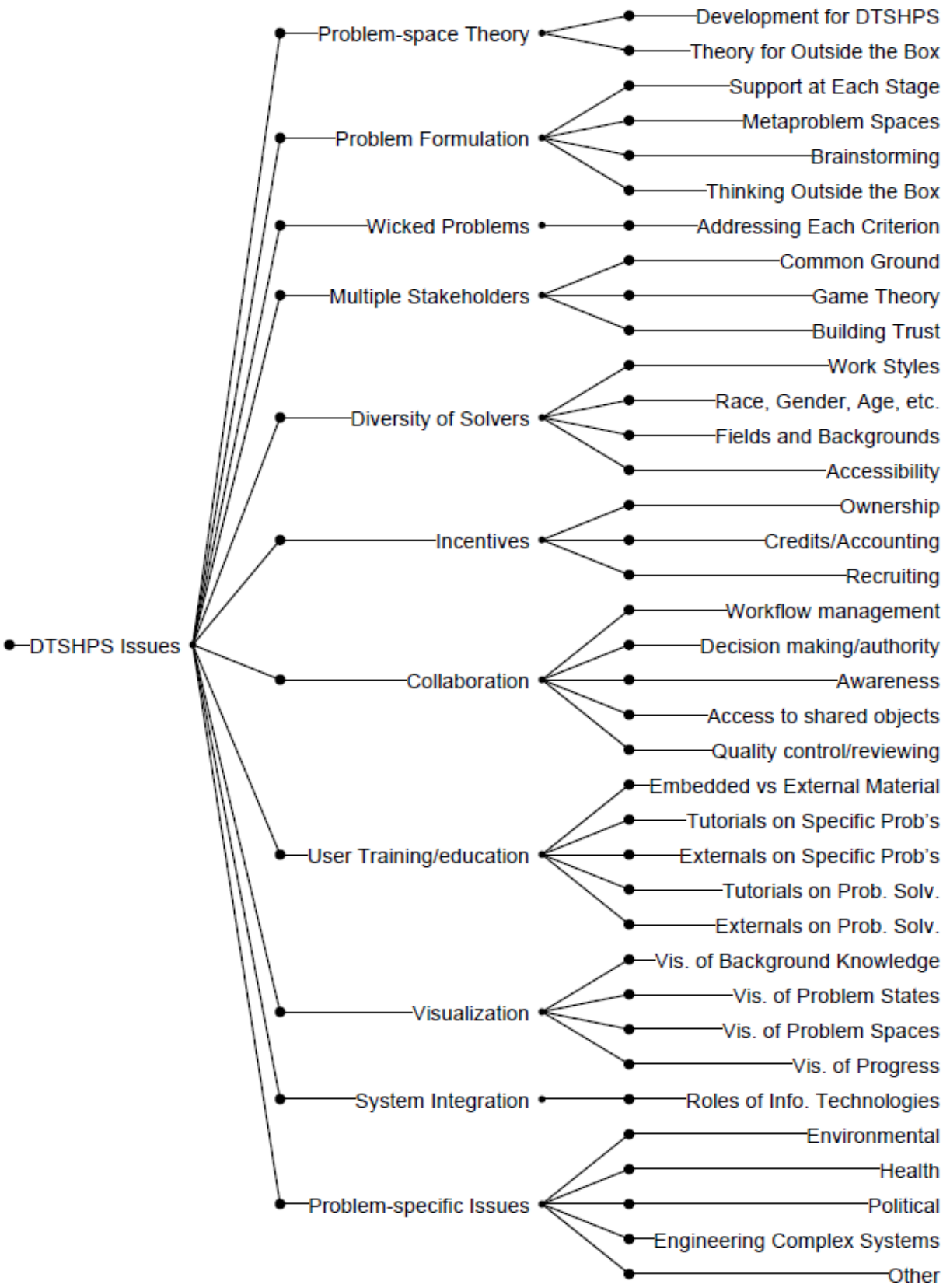

Fig. 1. Taxonomy of issues for designing technologies to support human problem solving. Each node of this tree is discussed explicitly in Section IV.

2) Ways to support "thinking outside the box" (TOTB): The classical theory of state-space search (see Simon and Newell [17]) may seem to be overly prescriptive. How can the theory accommodate human solvers who need to question assumptions and think outside the box?

Approaches include (1) the use of systems of "nested boxes" instead of one "box" (i.e., strict formulation), and (2) using constraints that can be dynamically added or removed in a problem formulation [9].

B. Problem Formulation Issues

1) Best ways to support the various stages of formulation: Here are key stages of the problem formulation process. For each stage there is an issue of how best to support users in that stage.

- identification of the need that corresponds to the problem,
- preformulation (resource discovery, retrieval and analysis),
- posing (decisions about what aspects are most important and relevant, as well as modeling the structure of a problem),
- coding (reduction to computer program snippets),
- testing and
- evaluation of a formulation.

Within the preformulation stage, the techniques of sense-making can lead to usable representations of the information relevant to the problem[15][13][11].

2) Metaproblem spaces: A metaproblem space for a given problem is a space of possible formulations of the problem [7]. The main issue here is how to exploit the metaproblem space notion, which can mean considering problem formulation as negotiating a trajectory through the metaproblem space. Metaproblem spaces can be formalized by setting bounds on possible formulations, such as limiting state representations to use particular kinds of program variables, such as integer variables, etc.

3) Brainstorming: Brainstorming refers to early-stage problem solving activity characterized by the free expression of as many ideas as possible, without critique or judgment. The intent is to prevent people from limiting their problem formulation or solution space by prematurely judging their own or others' ideas. How can we incorporate this aspect of brainstorming into problem formulation? Are there ways to create an environment that prevents critique in early-stage problem formulation?

4) How to accommodate "thinking outside the box": This notion seems most relevant when a problem has an existing formulation or an existing mindset associated with it which is so limiting that good solutions do not seem to be available. One way to encourage TOTB is to maintain a mindset that permits multiple formulations of any problem. If one formulation starts to seem too restrictive, then other formulations can be tried.

Another approach could be called "open formulations of problems" in which some parts of a formulation are considered variable and subject to alternative bindings.

C. Wicked problems

Articles on "wicked problems" typically cite Rittel and Weber's paper [14] when explaining the concept of wickedness. Ten properties of wicked problems were listed in their paper. These are:

1) There is no definitive formulation of a wicked problem
2) Wicked problems have no stopping rule
3) Solutions to wicked problems are not true-or-false, but good-or-bad
4) There is no immediate and no ultimate test of a solution to a wicked problem
5) Every solution to a wicked problem is a "one-shot operation"; because there is no opportunity to learn by trial-and-error, every attempt counts significantly
6) Wicked problems do not have an enumerable (or an exhaustively describable) set of potential solutions, nor is there a well-described set of permissible operations that may be incorporated into the plan
7) Every wicked problem is essentially unique
8) Every wicked problem can be considered to be a symptom of another problem
9) The existence of a discrepancy representing a wicked problem can be explained in numerous ways. The choice of explanation determines the nature of the problem's resolution
10) The planner has no right to be wrong

The central issue here is how to overcome each of these sorts of challenges. Possible approaches for each of the 10 criteria are the following.

(1) an acknowledgment that multiple formulations, with a good analysis of the strengths and weaknesses of each formulation, are more appropriate for wicked problems than a single, oversimplified, formulation.

(2) stopping rules may correspond to goal states in problem spaces, or they may correspond to criteria that determine when a solution is good enough. The concept of satisficing [16] addresses the issue of "optimal or good enough?"

(3) although some traditional problem-solving structures required "true-or-false"-like solutions (puzzles, logic questions, etc.), optimization theory and other methodologies readily accommodate not only fine gradations of solution quality, but multiple criteria.

(4) While no immediate or ultimate test for solutions may exist for wicked problems, rational evaluation criteria can be designed that serve as practical proxies for tests of success.

(5) The one-shot nature of solution opportunities for wicked problems is generally based on the assumption that no good-quality simulation software is available for the problem's domain. Today, simulation software is increasingly effective, and it can permit planners to test many possible solutions before making the large commitment of resources necessary to implement a solution to a wicked problem.

(6) The lack of enumerable potential solutions or permissible operations for a problem may be a consequence of a failure to formulate the problem appropriately. Also, at the time the

Dilemmas paper was written, computer programming practice was not as sophisticated as it is today.

(7) The uniqueness of each wicked problem, while a valid point, need not be an obstacle to principled solution techniques. There is an infinity of unique two-number addition problems, and yet each one of them is solvable by the same simple method of long addition. Qualitative uniqueness is more difficult to deal with, but then patterns do emerge among complex problems; for example, many global challenge problems have to do with resource distribution–food to people, clean water to people, health care to people, etc., and the uniqueness of each problem does not mean uniqueness in every respect.

(8) The fact that a wicked problem may be a symptom of another problem, is a way of saying that the underlying causes of a problem may be initially hidden or considered out of scope. Clearly, a good formulation must take into account all the available variables that might affect the ability to solve the problem.

(9) The alternative representations of a wicked problem relate to the need for multiple formulations, as mentioned for criterion #1.

(10) When Rittel and Webber wrote that "The planner has no right to be wrong" the government policy planning context probably put unreasonable expectations on what professional planners could deliver, given the tools they had. Today, an enlightened city council can engage community representatives and professionals that can show the results of simulations that account for uncertainty using probabilistic models. The planners then have a right to be wrong in the sense that an implemented solution may not turn out perfectly and under budget, but may fall within the range of anticipated possible outcomes.

Thus an important set of issues regards finding, implementing, and evaluating means of "taming" wickedness by addressing each of the wicked-problem properties. Such means can include various forms of simplification and stating of assumptions, and the incorporation of simulations in the problem-solving environment.

D. Multiple Stakeholders

1) Approaches to common ground: In one model, each party has a hierarchy of objectives. If these are sufficiently compatible, then common subgoals can be used as areas of cooperation, and hopefully the existence of some of these makes it possible to bring various stakeholders to the table.

2) Game-theoretic approaches: By recognizing instances of Nash equilibria and prisoners dilemmas, steps may be taken to ameliorate their negative effects on reaching globally optimal states.

3) Building trust: How to engender trust is a key issue when the stakeholders to a problem are in conflict. An important question is how to create protocols that scaffold the building of trust, or that reduce the need for trust while permitting cooperation to move forward.

E. Diversity of Solvers

A general sociological issue is encouraging diversity in solver communities. Breadth of perspectives can be valuable for solving when the knowledge, techniques, or motivations to solve a problem might be insufficient if the solving team is too homogeneous. Another reason to support diversity in problem solving teams is to enable a broad sense of ownership in a resulting solution that might actually affect the lives of these same or similar people.

Dimensions of diversity relevant to problem solving therefore include (a) knowledge of the problem-domain, (b) problem-solving skills, (c) motivation (ideally at least one team member will be motivated to address each element of the problem), (d) sociological group, including age, gender, race, ethnicity or religion. On the other hand, all parties in a solving team must be sufficiently compatible that they can collaborate effectively. They must either speak/write in a common language or have sufficient language interpretation services that they do not misunderstand each other too frequently.

1) Supporting alternative work styles: Luther and Bruckman found that online work groups tasked with creating original movies were difficult to lead [10]. Part of the difficulty stemmed from the artist culture of wanting to release only perfected work, and artists wanting ownership in the product of their work. On the other hand, with the advent of web programming, much of software engineering focuses on a collaborative launch-and-iterate approach. The general issue is how to accommodate such differences in work preferences.

2) Avoiding biases related to gender, race, and other factors: Automated agents today exhibit some troubling biases. A case in point is the failure to distinguish images of various primates from people. One way to avoid the bias is to remove the technical features that exhibit the bias. In the meantime, researchers are trying to reduce the degree of bias in agents [18].

3) Designing to encourage beginners and participants from varying fields and backgrounds: How can tools and systems encourage new solvers? By providing specific roles that newcomers can easily fill, some of the perceived barriers to participation can be removed [4].

4) Making problem-solving support tools accessible to people with disabilities: Recent work in the design of coding interfaces for visually impaired programmers suggests that new interfaces to support blind problem solvers could be created [12]. Special features to help visually-impaired users locate blocks on a screen might be a key part of such an interface.

F. Incentives

Successful collaborations require that all participants have incentives to participate. Participants may be intrinsically motivated to solve a problem that they personally care about, or because they wish to learn more about a problem, but there can also be extrinsic motivators such as monetary incentives (hourly pay, prizes for winning accomplishments, lotteries,

etc.) or social recognition. Some of the subissues relating to incentives are ownership, credits/accounting, and recruiting.

1) Ownership: The results of a problem solving activity may include solutions, partial solutions, constructed objects, or curated information gathered from the Internet. Intellectual property rights related to collaborative problem solving systems need to be spelled out, so that participants feel that they are being treated fairly by the community they join, and by the system.

2) Credits/Accounting: The credit that a person receives from work performed may be monetary or handled in other forms. If it is accounted for numerically, then appropriate mechanisms must be in place. They should be transparent, so that a user knows how s/he is being treated. If credit is handled in a non-numerical way, such as by citing usernames within each object or document created or added to by a user, then the rules for awarding such bylines should be clear.

3) Recruiting: The quality of problem solving activity within a human-user system will clearly depend on the abilities and attitudes of its users. A system is more likely to be successful if it supports guidelines for recruitment. Recruitment may include discovery and communication with prospective users, means for describing needed skills and task requirements, as well as means for allowing a group of solvers to make pitches to potential new members. System support for provisional group members can enable one approach to recruiting.

G. Collaboration

The mechanics of group work involves another set of issues that are not unique to problem solving but are nonetheless important in such work, and collaboration in problem solving may require that standard techniques be adapted.

1) Workflow management: The particular ways that problem-solving tasks get broken into subtasks and shared among multiple human solvers are important in shaping the experience of the group. For example, the balance between independence and close coordination may need to be adjusted both for the problem being solved and the preferences of the group members.

2) Decision-making/authority: Social structure within a team may be flat or hierarchical. Systems may support various structures and provide tools for voting, reaching consensus, representing group structures, etc.

3) Awareness: In group collaborations, it can be helpful for each user to be aware, to some degree, of other users' activities. For example, what other users are logged in, what work they've done, what they are working on, and any events that might affect one's own activities during the session. How such affordances such as notifications are shown and managed can be important in maximizing productivity and satisfaction.

4) Access to shared objects: In a system such as CoSolve, it is easy to imagine a situation where a solver is working on a formulated problem, but a poser is editing the formulation. There are many possible cases in problem solving where two users may wish to edit the same object but inconsistencies need to be avoided. Flexible mechanisms are needed that permit users to be maximally productive.

5) Quality control/reviewing: Mechanisms are needed to help team members evaluate their own work as well as the work of their teammates. Evaluations may involve technical tools for testing and validation as well as qualitative descriptions.

H. User training and problem-solving education

Because solving complex problems may require a great deal of information about a problem as well as about how to solve complex problems, a suitable problem-solving system must either internally provide tools for learning to problem-solve, or it must rely on external training resources.

1) Embedded tutorials vs external courses.: A first question to ask when developing the learning materials is what parts of the tutorials or resources to include inside the system and what parts to make external. Internal learning aids can be more closely integrated with other interface tools or problem elements. External aids are easier to provide, they can have varying formats, and they may be easier for new users, who don't yet know how to access internal resources, to access.

2) Embedded tutorials about particular problems: How should such tutorials be a part of the system? Assuming that the tool is domain-independent but the sessions and posed problem are domain-dependent, links from the session to external tutorials that are problem-specific should be created by posers and lead solvers. Such links could be embedded in comments on objects within the posed problem or within a solving session. They could also be inside a chat history that is attached to the solving session.

3) External materials about particular problems: External materials may require some kind of cross-referencing within the system, so that users can transition easily between internal situations and the external resources. How should these links work and what can a system designer do to facilitate their creation and use?

4) Embedded tutorials about problem solving and the use of the tools: One of these could simply be an integrated help system for the tool. Possible technical affordances that facilitate tutorial integration are location codes (within solving-session objects), context snapshots, and the automatic detection of teachable moments when a pattern of user activity indicates that a particular lesson might be pertinent. Another possibility is for the system to help lead users or instructors to construct lessons from "real world" experience – that means work already done by the learner-user or by others on solving the current problem.

5) External materials about problem solving and collaborative approaches: More general educational background on problem solving and collaboration might be better provided using external resources such as Youtube videos, articles, books, and exercises.

I. Visualization

Whereas a fully automated solving agent using a computer algorithm to solve a pre-formulated problem might not have

any use for a visualization of any kind, when human solvers are involved, visualizations become essential. Human understanding of a complex problem space can be greatly helped using appropriate visualizations.

1) Visualization of Background Knowledge for a problem: Maps, charts and interpretive diagrams can help new users become familiar with problem background. Visualizations can be particularly helpful during problem formulation when collaborative sensemaking is used to analyze information resources and develop problem representations [11]. The use of large, high-resolution displays can facilitate such processes [1].

2) Visualization of problem states: A problem state typically represents the arrangement of elements of a solution after some steps towards the solution have been taken. Portraying the state of a problem with diagrams such as plots and charts, or image renderings, can be very helpful in explaining what has been accomplished.

3) Visualization of problem space: The entire space of possible states of a problem can sometimes be shown visually, although often in greatly simplified form. Such a display can help users understand the relationships among various states. Here, graph layout, choices of axes, projection and remapping, infinite graphs, and interactive display of graphs are relevant techniques.

4) Visualization of the work done to solve a problem.: As a group works on a problem, simply keeping track of the group's progress becomes nontrivial. Relevant techniques include maintaining and showing session histories (e.g., with tree diagrams), showing the best path through state space found so far, and plotting of landmarks (in the problem space) created by members of the solving team.

J. System Integration

1) Roles for information technologies: This is a broad issue, and we are lumping together the many different information technologies so that we have space to call out the issues above.

Where do various information technologies fit into a system that supports collaborative problem solving? Particular technologies include numerical algorithms, artificial intelligence techniques (logical inference, pattern recognition, language understanding, recommendations, searches, etc.), machine learning, collaborative editors, chat and conferencing, version control, visualization, database management, user account management, security, and others.

K. Problem-specific Issues

Particular problems or categories of problems may have somewhat unique issues associated with them. For example, environmental problems typically have a geo-spatial aspect that may call for an integration with geographic information systems (GIS). We mention a few categories here to illustrate the general issue.

1) Environmental: Such problems include global climate change, ocean acidification, sea-level changes, air pollution, water supply. As mentioned above, these share a geo-spatial aspect.

2) Health: Drug-resistant diseases, affordable health care, health education. Such problems share a possible need for a system of human values related to health care, such as the notion that access to basic health care is a human right.

3) Politics, War, Nuclear Proliferation: Problems related to violence and war have their own issues including the human costs of suffering, deaths (due to small-scale crime, genocide, or apocalyptic war) and systems of human beliefs.

4) Engineering of Complex Systems: Software and cyber-physical systems can be sufficiently complex that some of the problem-solving techniques referred to earlier are appropriate. Debugging alone can require problem-solving skills and tools.

5) Other: Fake news is just one example of a problem that does not fit well into the four categories above. It has its own set of issues related to the nature of truth, and the goals of various parties such as news media, and power brokers.

V. Research Prioritization

This section considers the relative importance of some of the issues discussed earlier.

A. What issues are most important?

The answer here depends on several factors, including which problems need to be solved first and who is ready to engage in solving them, as well as what tools are being considered for modification or a complete design and build.

1) Prioritization of Problems: The relative importance of problems themselves can be considered from several points of view, such as time frame, numbers of people affected, and available opportunities. Time frames run from short-term (needing solutions very soon) to long-term (needing solutions a few years from now). Averting an impending asteroid strike is a problem that may come with a very accurate deadline. Slowing global warming is perhaps just as pressing but without such a clear deadline. Achieving a culture of problem solving through educational curriculum changes may be very desirable, but it seems to fit into a long-term time frame.

Problems that affect larger numbers of people may be considered as higher priority problems.

B. Finding frameworks that can be used to integrate services and solutions to the issues

If existing tools are to be extended to address particular issues identified here, then the existing tool may impose the framework for such additions. Tools such as Jupyter notebooks, SAGE math sessions, CoSolve, or automatic solvers that work with PDDL or its variants [19] are possible candidates for extension, but there are doubtlessly others.

REFERENCES

[1] Christopher Andrews, Alex Endert, and Chris North. “Space to think: Large, high-resolution displays for sensemaking.” Proc. CHI 2010. Atlanta, GA.

[2] Jeffrey Conklin, Jeffrey. *Dialogue Mapping: Building Shared Understanding of Wicked Problems*. Chichester, England: Wiley Publishing. ISBN 0-470-01768-6. 2006.

[3] Sandra B. Fan, Tyler Robison, and Steven L. Tanimoto. ”CoSolve: A System for Engaging Users in Computer-Supported. Collaborative Problem Solving.” Proceedings of VL/HCC 2012, Innsbruck, Austria.

[4] Sandra B. Fan. *CoSolve: A Novel System for Engaging Users in Collaborative Problem Solving*. Ph.D. dissertation, University of Washington. 2013.

[5] Álvaro Figueira and Luciana Oliveira. “The current state of fake news: challenges and opportunities.” https://doi.org/10.1016/j.procs.2017.11.106

[6] Tim Gowers, “Can Polymath Be Scaled Up?” 2009. [Online]. Available: http://gowers.wordpress.com/2009/03/24/can-polymath-be-scaled-up/

[7] John Jackman, Sarah Ryan, Sigurdur Olafsson, and Veronica J. Dark: “Metaproblem Spaces and Problem Structure” in D. Jonassen (ed): *Learning to Solve Complex Scientific Problems*. Lawrence Erlbaum, 2007.

[8] Koios.com website. Accessed 13 July 2018.

[9] Denis Lalanne, 1999. *Conception créative assistée par ordinateur: un modèle d’intelligence interactive*. Ph.D. dissertation. (in French).

[10] Kurt Luther and Amy Bruckman. “Leadership in online creative collaboration.” In Proceedings of the 2008 ACM conference on Computer supported cooperative work (CSCW ’08). ACM, New York, NY, USA, 343-352. 2008.

[11] Narges Mahyar. *Supporting Sensemaking during Collocated Collaborative Visual Analytics*. Ph.D. dissertation, University of Victoria, BC, Canada. 2014.

[12] Lauren Milne. *Touchscreen-Based Learning Technologies for Children who are Blind or Have Visual Impairments*. Ph.D. dissertation, University of Washington. 2018.

[13] Sharoda A. Paul and Madhu C. Reddy. “Understanding together: Sensemaking in collaborative information seeking.” In *Proc. ACM Conf. on Computer Supported Cooperative Work*, pp.321–330. ACM, 2010.

[14] Horst W. J. Rittel, and Melvin M. Webber. “Dilemmas in a general theory of planning.” *Policy Sciences* (June 1973), Vol. 4, Issue 2. pp.155-169. https://doi.org/10.1007/BF01405730.

[15] Daniel M. Russell, Mark J. Stefik, Peter Pirolli, Stuart K. Card. “The Cost Structure of Sensemaking.” *Proceedings of INTERCHI’93*, pp.269-276.

[16] Herbert Simon. *The Sciences of the Artificial*. Cambridge, MA: MIT Press. 1969.

[17] Herbert Simon, and Allen Newell, “Human Problem Solving: The State of the Theory in 1970.” *American Psychologist*, 1971, pp.145-159.

[18] Tom Simonite. “When it comes to gorillas, Google Photos remains blind.” https://www.wired.com/story/when-it-comes-to-gorillas-google-photos-remains-blind/. 2018.

[19] Volker Strobel and Alexandra Kirsch. “Planning in the Wild: Modeling Tools for PDDL”. In *KI 2014: Advances in Artificial Intelligence, Proceedings of the 37th Annual German Conference on AI*, Stuttgart, Germany, September 22-26, 2014.