End-User Development in Internet of Things: We the People

Margaret Burnett
Oregon State University
Corvallis, OR 97331 USA
burnett@eecs.oregonstate.edu

Todd Kulesza
Oregon State University
Corvallis, OR 97331 USA
kuleszto@eecs.oregonstate.edu

Abstract
This position paper considers people aspects of end-user development in the Internet of Things.

Author Keywords
End-User Development, End-User Programming, End-User Software Engineering, Gender HCI, Internet of Things

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

Introduction
The Internet of Things is inherently about people—helping to improve their safety or their experiences in their homes, jobs, and in the world. For example, we can warn people of dangerous situations near their current location, recommend the next painting to look at in a museum, or enable them to remotely "program" objects in their smart homes to talk to each other.

My interest is in the "programming" aspect of this, specifically: (1) how we can enable ordinary users to customize, control, and "fix" Internet of Things applications that are trying to help them, and (2) doing so in ways that allow IoT systems to be inclusive to both males and females.

Meet "Grandma"

Imagine "Grandma". Grandma has lived in the same small town and even the same house for her entire adult life. Even though she is advancing in years, she does not want to leave her home to live with relatives or in a nursing home.

Her granddaughter, Mary, would be happy to have Grandma come live with her in Los Angeles (about 100 miles from Grandma's home), but Grandma will have nothing to do with this plan. She used to be a teacher, and her former students still recognize her when they run into her at the neighborhood grocery or library. She is very involved in her church, her neighborhood, and her community. In short, she feels like she is where she belongs, and moving would make her feel like she was giving up most of what she cares about.

Together, they decide on an aging-in-place solution, which uses IoT technology. Using this approach, smart technologies around Grandma's home can alert Mary when Grandma's activities seem abnormal, or when safety issues around the house arise, like water overflowing from a forgotten faucet, smoke, possible trespassers, and so on. Mary can even control some of the technologies remotely, like turning off the faucet.

End-User Customization, Control, and "Fixing" of IoT Mis-steps

However, smart technologies are never 100% correct, and some of the mistakes the system makes can be very costly to Mary, and even dangerous to Grandma. For example, Mary does not want a lot of false positives from the system, because this will mean a 100-mile drive one-way to Grandma's, only to find out, for example, that the system's warnings of intruders oftentimes are just branches blowing against the house. And Mary does not want false negatives, because she cannot risk leaving Grandma in a precarious or dangerous situation simply because the system thinks the situation matches patterns it has seen before.

Mary needs to fix such false positives and negatives, but how? Grandma's data is in many ways unique to Grandma, so Mary cannot rely on other grandmas' data to train the system better. She needs some amount of direct control.

We have been working on an approach to help Mary exert this kind of control. We call the approach "Explanatory Debugging" [5]. The approach aims to enable end users like Mary to efficiently influence the predictions that machine learning systems make on their behalf, such as the system powering Grandma's house. This paper presents an overview of Explanatory Debugging, an interactive machine learning approach in which the system explains to users like Mary how it made each of its predictions, and the user (Mary) then explains any necessary corrections back to the learning system.

The Explanatory Debugging approach presents explanations that people like Mary can then act upon. These explanations need to honor two sets of principles: a set of Explainability principles and a set of Correctibility principles:

- Explainability:
  - Principle 1.1: Be *iterative*, so that the user can learn in increments via many interactions.
  - Principle 1.2: Be *sound*; explain the system truthfully.
  - Principle 1.3: Be complete; explain how the *entire* system operates.
  - Principle 1.4: But don't overwhelm; if the system is too complex to explain, perhaps a different machine-learning model would be more appropriate.
- Correctibility:
  - Principle 2.1: Be actionable.
  - Principle 2.2: Be reversible.
  - Principle 2.3: Always honor user feedback.
  - Principle 2.4: Incremental changes matter.

For example, Grandma does not like the sensors in the bathroom to be on when she is actually in there, so she covers them up with a swatch of dark material when she uses it. This causes the system to raise alarms, but Mary and Grandma talk it over, and Mary decides that as long as the bathroom sensors are not "dead" for more than 30 minutes, she will not worry about it.

She needs to explain this to the system in terms that will not work at cross-purposes to the system's ways of reasoning; this is where Principles 1.2 – 1.4 are especially important. (If the system does not clearly explain how it is working, how can Mary understand how to correct its behavior?) Further, she needs to know what pieces of the explanation she should change (Principle 2.1), be able to tell whether her corrections are on the way to producing the right behaviors (Principles 1.1, 2.3, and 2.4), and be able to undo corrections that did not work out well (Principle 2.2).

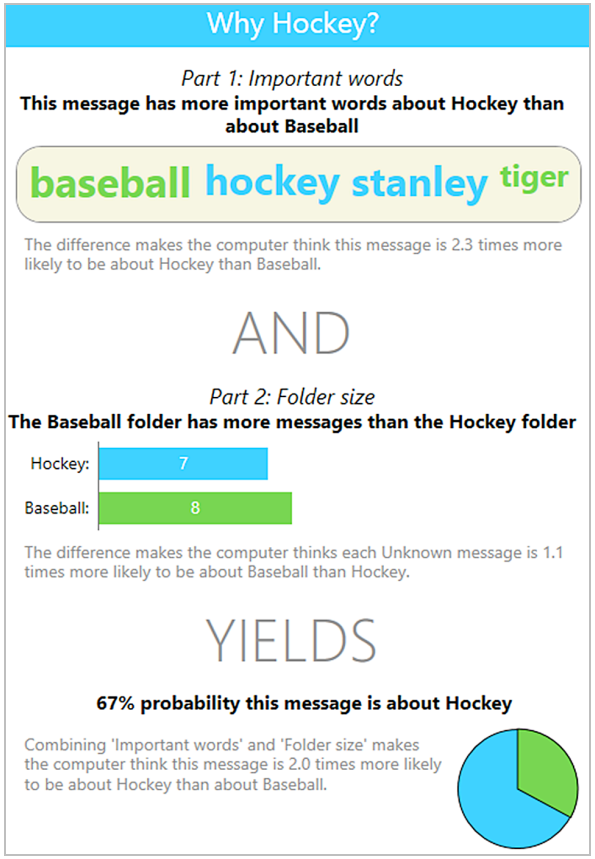

Figure 1: An Explanatory Debugging [5] "why" explanation: It tells users how features and folder size were used to predict each message's topic.

One example of Principles 1.2 – 1.4 in a "baseball vs. hockey" text classifier is shown in Figure 1 [5].

Gender Inclusiveness

Mary is, obviously, a female. Will the emergent Internet of Things systems fit her style of working and problem-solving as well as they fit that of male adult caregivers with aging-in-place grandmothers?

Judging from the recent past, the answer is likely to be no. Recent research has shown that the way people use software often differs by gender, and further, that many software features are inadvertently designed around the way males tend to work and problem-solve with software [1, 2, 3, 4, 6, 7].

These findings show the importance of taking gender differences into account when designing software for solving problems, such as the software Mary is using in the example above. Fortunately, design remedies are emerging to overcome gender-inclusiveness issues, and such changes need not trade off one gender against the other. In fact, researchers have shown that taking gender differences into account in designing software features can benefit both genders. For example, Tan et al. showed that displaying optical flow cues benefited both females and males in virtual world navigation [7], and Grigoreanu et al. showed that changes to spreadsheet features relating to confidence and feature support reduced gender gaps while improving both genders' attitudes and feature usage [3].

Conclusion: Don't Forget the People

We hope that this new generation of software powering the Internet of Things will be as helpful to ordinary people as it is to corporate interests, engineers and tech-geeks, and equally inclusive to females and males.

Acknowledgments

This work was supported in part by NSF #1240957 and #1314384.

References

[1] Burnett, M., Fleming, S., Iqbal, S., Venolia, G., Rajaram, V., Farooq, U., Grigoreanu, V., Czerwinski, M. Gender differences and programming environments: across programming populations. *ACM-IEEE Empirical Software Engineering and Measurement (ESEM)* (2010), Article no. 28, 10 pages.

[2] Chang, S., Kumar, V., Gilbert, E., Terveen. L. Specialization, homophily, and gender in a social curation site: Findings from Pinterest, In Proc. *Computer Supported Cooperative Work & Social Computing*, ACM Press (2009), 674–686.

[3] Grigoreanu, V., Cao, J., Kulesza, T., Bogart, C., Rector, K., Burnett, M., Wiedenbeck, S. Can feature design reduce the gender gap in end-user software development environments? *IEEE Symposium on Visual Languages and Human-Centric Computing* (2008), 149–156.

[4] Kulesza, T., Stumpf, S., Wong, W., Burnett, M., Perona, S., Ko, A., Oberst, I. Why-oriented end-user debugging of naïve bayes text classification. *ACM Transactions on Interactive Intelligent Systems*, 1(1) (2011).

[5] Kulesza, T., Burnett, M., Wong, W.-K., Stumpf, S. Principles of explanatory debugging to personalize interactive machine learning, *Proc. IUI*, ACM (2015).

[6] Piazza Blog, STEM Confidence Gap (2015). http://blog.piazza.com/stem-confidence-gap/

[7] Tan, D., Czerwinski, M., Robertson, G. Women go with the (optical) flow, In Proc. *CHI*, ACM Press (2003), 209–215.