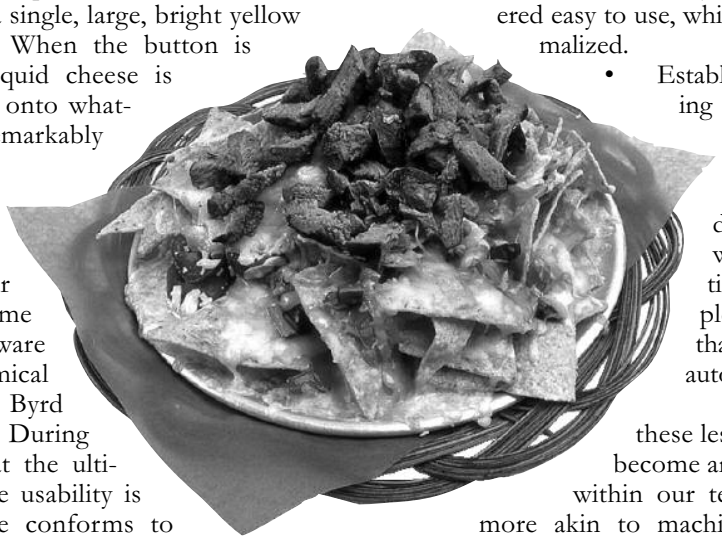
Push for Cheese: A Metaphor for Software Usability

At the National Radio Astronomy Observatory's (NRAO) Science Center in Green Bank, W. Va., visitors curious about radio astronomy and the observatory's history and operations will discover an educational, entertaining experience. Employees also visit the science center, but their thoughts are more on afternoon snacks rather than distant galaxies. The employees of NRAO's Software Development Division in Green Bank have gained tremendous insight on the topic of software usability from many visits to the Science Center Café by pontificating upon the wisdom inherent in the design and use of the liquid cheese dispenser there.

The cheese dispenser is used to coat nacho chips in the familiar orange-tinted, viscous plasma prior to the addition of jalapeno peppers (or other toppers), intended to enhance the consumer's experience of the food product. The user interface is clear and unambiguous, with a single, large, bright yellow button labeled *Push for Cheese*. When the button is depressed, a stream of hot liquid cheese is expelled from the machine and onto whatever lies below. The cheese is remarkably consistent in its appearance, texture, and temperature. There is always only one variety of cheese dispensed.



Over the past three years, our team has spent considerable time and effort to improve the software systems used to make astronomical observations with the Robert C. Byrd Green Bank Telescope (GBT). During this time, we've discovered that the ultimate measure of GBT software usability is how well the user's experience conforms to what happens when they Push for Cheese. Our users want to press one button, have the software automatically interpret what they want the telescope to do, then see their results presented in a comprehensive, straightforward way. Though it's a lot to ask from software (especially since there are thousands of ways the GBT can be configured), similar concepts have been envisioned for years: In 1950, Turing argued that within five decades, computers would be intelligent [1]. In 1990, Newell presented his vision of a fully networked, intelligent environment in which machines and humans seamlessly interact (ubiquitous computing) [2]. At least in our environment, the knowledge embodied in the software is closely related to its perceived usability.

The International Organization for Standardization 9241 Part 11 defines usability as the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" [3]. Implicit in this definition is the notion of expectations because machines (and software) will only express the intelligence we seek. Who are our specified users? What is the context of use? How can we define and measure satisfaction? Here's what we've learned:

- Our users ultimately want a Push-for-Cheese user interface, where minimal and straightforward interactions with the software achieve the desired result.
- Conversely, users don't want the software to make too many assumptions on their behalf. As a result, more options may be required to achieve the balance between ease of use and software that's *too smart for its own good*. Push for Cheese assumes that the consumer wants liquid nacho cheese, which is a fine assumption for nachos, but not for a grilled cheese sandwich.
- Expectations are critical. Additional options can be designed into the system, but to add them, we must first establish requirements then develop, test, and deploy, which requires time and effort. Anyone who expects Brie or Swiss on their nachos when they Push for Cheese will be disappointed unless we've planned for other cheeses in advance.
- Even within a well-defined segment of specified users, there will still be substantial subjective variation in what's considered easy to use, which must be aggregated and normalized.
- Establish usability requirements during the analysis stages of a project to set bounds. A user expecting a graphical user interface won't like a command-line driven program; similarly, one who wants to specify many settings before execution will not be pleased with *intelligent* software that selects and tweaks settings automatically.

In addition to illustrating these lessons, Push for Cheese has also become an effective way to communicate within our team. When a usability feature more akin to machine intelligence or ubiquitous computing is requested, and is posited to us under the guise of ease of use, we console each other by affirming that they just want to Push for Cheese. In doing so, we compassionately acknowledge their desires, and then move to meet them halfway in implementation – while keeping the Push-for-Cheese vision for usability close to our hearts.

— Nicole Radziwill
— Amy Shelton
National Radio Astronomy Observatory
nradziwi@nrao.edu

References

1. Taatgen, N.A. "Poppering the Newell Test." Netherlands: University of Groningen, 2003 <www.ai.rug.nl/prepublications/BBSAlcom-NT-2003.pdf>.
2. Gray, J. 1999: "What Next? A Dozen Information Technology Research Goals." MSR-TR-99-50. Redmond, WA: Microsoft Research, June 1999 <http://research.microsoft.com/research/pubs/view.aspx?msr_tr_id=MSR-TR-99-50>.
3. International Organization for Standardization. "ISO 9241 Part 11: Guidance on Usability." Geneva, Switzerland: ISO, 1998 <www.userfocus.co.uk/resources/iso9241/part11.html>.