Contents

Abstract

Constructing hypothetical scenarios and user naratives is a common technique for communicating the envisioned user experience (UX) of a tool. Using this approach, application developers rapidly build stories of expected use cases that concretize the intended UX. This design strategy is effective, cheap, flexible, and simple to implement. However, most narrative scenarios are not formal, rather they lack a mathematical basis suitable for statistical testing and rigorous delineation of usership patterns. In this paper, I describe an effort for underpinning use case scenarios (UCSs) with a probability framework. Under this framework, UCSs are modified to include probability distributions describing the use of each interface component. This approach offers at least three advantages that complement traditional scenario-based design approaches: (1) it enables formal statistical testing of use patterns, with both hypothetical data and, once the tool is put into production, real usage data, (2) it creates the potential for novel visualizations of application use, and (3) it creates the possibility of incorporating Bayesian data assimilation into interface design decisions, offering insight for future designers. Moreover, the method provides a way of informing personalization and customization traditional in large e-commerce systems with much less data, ideal for systems with fewer users. Here, I illustrate the application of the framework using a conceptual example and a hypothetical interface. Several UCSs are explored and fuzzy statistical methods are shown to effectively delineate between them when configurations are drawn at random from the newly implemented probability distributions.

Introduction

Powerful web programming frameworks and widespread consumer access to low-cost, high-speed, internet-enabled computing devices have resulted in a host of highly interactive, richly-featured applications that use the Web as a platform. These apps rely on a two-way communication model that encourages user-generated content generation and social interaction between users. As cloud services gain popularity, these web apps are an increasingly large portion of the software used on a daily basis. Workflows, such as email and calendar, and increasingly, work activities, such as word processing and data analysis, are done on software 'in the cloud'. That is, these applications are hosted on cloud-based infrastructure, forming a software-as-a-service (SaaS) that enables synchronous information retrieval and preference persistence across devices, creating a seamless user experience. Often, these applications serve multiple user groups, with different interests, motivation, or skills. To maintain a positive user experience for all users, artificial intelligence (AI) and machine learning (ML) algorithms are often applied to recognize user actions, identify likely sequences of interactions, recommend suggested products, and adapt the interface to the individual's preferences. These approaches improve the design of web-based systems by exposing the user to less information and visual clutter, allowing their cognitive function to be more focused on the reason for using the interface in the first place. While AI and ML personalization and customization approaches are well-theorized, widely used, and generally accurate, these techniques have two important drawbacks. First, because they typically model the user based on the sequence of interactions taken by a user during an interaction session (i.e., the clickstream) these algorithms often require large amounts of data from many user sessions. Such large amounts of real usage data may not be available for alpha- or beta- stage applications still under active development. Second, AI approaches do not typically account for expected usage as envisioned by the application developers and domain experts. In many cases, these stakeholder groups go to great lengths to characterize the user and their expected user experience (UX) with the interface.

Specifically, application developers often work with a team of designers and domain experts to develop narrative scenarios about the expected UX of a tool in a process known as scenario based design(J. M. Carroll, 2000; SBD, M. B. Rosson, 2002). Scenarios provide a common vocabulary for communication between stakeholders by describing how a user will interact with a tool (J. M. Carroll, Rosson, Chin, & Koenemann, 1998). SBD is used in a wide variety of fields, and is not limited to development of software systems (Go & Carroll, 2004; Kazman, Abowd, Bass, & Clements, 1996). Within software development, it is often used to inform the requirements of a proposed tool during their negotiation. Scenarios provide a flexible and cheap method of communicating concrete use cases, and have been shown to improve utility and usefulness of the resulting tool(J. M. Carroll et al., 1998).

However, by definition, the scenarios produced during SBD are informal (M. B. Rosson, 2002; M. B. Rosson & Carroll, 2005). If a scenario is formalized, it changes in both form and purpose, from an illustrative device for communication to a document describing the functional requirements of the tool (Go & Carroll, 2004). Requirements documents are often more concerned with the feasibility of the tool than the utility and usability by the envisioned user. In the present study, I develop a method, probabilistic scenario based design (pSBD), for creating formal user scenarios that maintain the connection to the envisioned users and their intended use of the tool under development by enhancing traditional SBD scenarios with probability distributions. In this method, each scenario consists of probability distributions that describe the use of each proposed interface component, function, or logical element in the envisioned interface. In a simple case, these distributions can simply represent the probability that the user described in the scenario will use that component. In more complex cases, these distributions can describe the dimensions of a user-configurable layout or the geographic center of a map component.

This development model has advantages for UI developers and could alleviate some of the challenges associated with implementing AI-based personalization systems. By introducing a probability model, pSBD facilitates improved inference of user interaction patterns, before the interface has undergone extensive use. Rather than informally constructed narratives, scenarios become suitable for formal statistical testing of user interaction patterns, distinction between real usage and envisioned usage, and prediction of success of future interfaces at a component level. Because the probability model is developed during the planning and development stages along with the UI, AI customization systems have data to work with immediately. And, because a probability distribution already describes the application developers' best estimate of UI usage, less data is required to determine if a future user falls into a particularly user model.

While AI algorithms may require significant modifications to work within the pSBD framework, I imagine at least three immediate and important benefits of pSBD, illustrated in the case study below. First, this method would allow for novel visualizations techniques. These visualizations would show in a concise manner how target user groups' UX would differ, thereby improving communication between stakeholders, developers, and designers. Second, pSBD enables the formal statistical testing of interaction with an interface. Specifically, pSBD allows the testing of if real usage patterns conform with the expectations of the development team. This could be particularly useful during design interations to improve the interface to meet expectations. Finally, pSBD is amenable to Bayesian data assimilation. Observations of actual usage can be incorporated into the probability framework as the application is deployed. The interface can then adapt, in an intelligent way, to reflect both the envisioned usage of the designer and the real world usage patterns of the target audience. Furthermore, these findings can be formally shared to improve the design of future interfaces by providing an informed set of priors for use in future scenario development.

The balance of this paper proceeds as follows. First, I outline contemporary techniques in user interface personalization and recommendation. I argue that that SBD provides an intriguing opportunity to integrate additional information into the process of personalization systems. Additionally, I discuss existing methods for including Bayesian inference and prediction in the design and function of user interfaces. Second, I describe the methodology for enhancing simple scenarios from SBD with probability distributions, with special consideration on techniques for negotiation the distribution between developers, designers, and domain experts. Third, I illustrate the derived statistics, visualizations, and prediction possible with the new method. Finally, I conclude with by addressing the limitations of the methodology and its utility in the real world.

Background and Prior Work

Intelligent User Interfaces

With the ubiquity of modern computing, computer-based tool and their user interfaces are all around us. They facilitate listening to music, ordering food, getting direction, following current events, and chatting with friends. As Schniederman suggests, well designed computer interfaces positive feeling of success and competency. If designed correctly, the interface recedes, enabling users to focus on their work, exploration, or pleasure (Shneiderman, 2010). Artificial intelligence (AI) plays a big role in helping the interface to disappear. AI recommends products that are similar to products already purchased by a consumer (G Smith Linden B.; York, n.d.). In general, AI recommender systems can improve user interfaces in two ways. First, they can expose users to less choice. For example, ather than browsing a catalogue of ten million products, an ecommerce store can expose users to only ten that were purchased by similar consumers. Second, they can improve user interface interactions by partially executing a task based on knowing that a user belongs to a group that typically executes that task.

In both cases, user modeling adapts the interface and the interactions needed to improve the user's session with the application, based on characteristics of the user. Specifically, user modeling is the process of characterizing the users of an application in a way that can be communicated or integrated into a personalization system. User modeling is a productive field of contemporary human-computer interaction (HCI) research, and its implementation is important in recommender systems, social computing, intelligent web search, personalized help systems, adaptive interactive systems and intelligent user interfaces (Cunha, Pereira, Gomes, Pereira, & Santos, 2014).
While algorithms and techniques for user modeling and personalization of web systems is typically associated with ecommerce, the sciences are increasingly relying on AI to improve scientific investigation. Researchers in many fields are currently leverage these algorithms to identify pertinent literature and in

visualization systems that help to identify anomalies. For example, the Sunfall system employs sophisticated algorithms and novel visualization techniques to facilitate deep scientific inquiry into vast amounts of data generated by a supernova search project (Aragon, Bailey, Poon, Runge, & Thomas, 2008). Sunfall incorporates usability principles and cognitive load considerations in the design of a visual analytics interface (Gil, Greaves, Hendler, & Hirsh, 2014). Other tools, real and envisioned, can sift through published scientific literature generating new synthetic knowledge in the process. For example, GeoDeepDive is an effort to search through existing literature to identify improve data extraction that currently indexes over 3.4 million published papers (Zhang et al., 2013). In general, with such systems scientists are able to focus on more of the creative aspects of research, while routine tasks are left to an intelligent agent (Gil et al., 2014).

There are two main categories of AI often used in the development of web-based interfaces. First are adaptive user interfaces (AUIs) that identify a user based on their actions and then automatically execute routines commonly executed by users conforming to that model. AUIs are systems that adapt to the needs of different users for a variety of tasks (Isbell & Pierce, 2005). AUIs are often implemented in the context of intelligent tutoring and educational systems, in the user model controls and keeps track of how a user is progressing towards an educational goal (Conati, Gertner, & Vanlehn, 2002). In general, AUIs work by identifying membership in a user group based on a series of events. In effect, this means analyzing the click stream from a user's interaction session with the interface. This can be a challenging amount of data to manage and analyze. p.3: To detect user behavior patterns, the interface must be able to track all basic interface events. – Highlighted May 2, 2017 p.4: Adaptive interfaces also need data from users after they act. The more proactive an adaptive interface, the larger the amount and quality of feedback it needs from users in order to determine whether or not it has acted appropriately and whether it should modify its behavior. – Highlighted May 2, 2017

A second key form of AI often found in web-based applications is the recommendation system. These systems were pioneered for use in e-commerce, at large online retailers, such as Amazon.com, but have recently come into use in many other fields (G Smith Linden B.; York, n.d.). {MORE ABOUT THIS}. Recommendation systems have been shown to significantly improve common e-commerce statistics, like conversion rate.

Predictive statistical models are often used to determine what a user will do, based on the sequence of actions they've taken during an interaction session, or, in the case of cloud-based applications, over the course of all interactions made with the interface. Most of these are focused on identifying the plan the user will most likely take, given that they have taken a certain sequence so far. These models are typically divided into two categories, content-based or collaborative [CITE] [MORE ABOUT THIS]. p.11: These enhanced models can then make more accurate predictions about the behaviour of individual users by matching

these users to a particular group. – Highlighted May 2, 2017

While AI is often used to improve sales in an online context, it has also been used in the past to improve the design of an application. Particularly in the context of AUIs, computing power can be used to determine how and when to proactively assist the user (Isbell & Pierce, 2005). Multi-level interfaces are an important target for AI assistance. For example, a novice user would receive a more detailed and longer sequence of dialog steps with additional extent of assistance [CITE]. This often requires checking whether the user has the required knowledge to complete the steps on their own. Some systems are able to adaptively configure themselves to display only the components most suited to the current user of the tools and their immediate goal. p.380: It determines an adequate combination of diverse user interface components, adapted to the identified context of use. – Highlighted May 2, 2017 p.380: Each addressed user interface component renders an individually assigned part of the output as final user interface (see Fig. 2). – Highlighted May 2, 2017

These techniques all typically require that the algorithm be provided with a tremendous amount of data. Specifically, these algorithms often need to know the entire click stream of a user's session to place them into a user model and make plan for what the user is likely to do next. Moreover, to train these algorithms, data on many users is required. Finally, while these technique reflect the actual use of the tool, the intended use of the tool, as characterized by the tool developers and domain experts, is not taken into account.

Scenario Based Design

Scenario based design is a quick and dirty way to communicate envisioned use and who is using a tool

SBD has been used in many contexts

SBD is widely used and provides concrete examples that are easy to communicate

SBD is qualitative – requirements are formal

PSBD could be a bridge between the two

Bayesian Inference

Bayesian networks are common in user modeling studies

Here, we don't propose a conditional probability bayesian network – that would be difficult to construct by hand

Thought for future research: Bayesian network for design

If we had PSBD, we could use Bayesian inference to see if our expectations are met by actual use We could improve our expectation for future interfaces, even if we don't build the same interface again (better priors)

Bayesian data assimilation is often used in weather forecasting, etc

Could extend this to a multivariate case, right now its a simple univariate case

Therefore,

PSBD would

Limitations

Predictive user model (plan making) AI algorithms assume that users maintain their preferences over time p.10: An implicit working assumption of many predictive models is that of `persistence of interest` (Lieberman, 1995), whereby users maintain their behaviour or interests over time. – Highlighted May 2, 2017

But, the group as a whole can change their preferences, and this system will reflect that paradigm shift p.10: An implicit working assumption of many predictive models is that of `persistence of interest` (Lieberman, 1995), whereby users maintain their behaviour or interests over time. – Highlighted May 2, 2017

Bayesian Assimilation in Interface Design

Case Study Introduction

Here, I introduce a hypothetical interface with two components and two target user groups as a case study with which to demonstrate my method.

Method

The method of underpinning a use-case scenario with probabilities of component use has two prerequisites. First, the application developer and stakeholders must work together to come up with a wireframe of the intended interface. A wireframe of the application, including all components, their function, and their position on the page is essential to this method. Wireframes can be either high-fi or low-fi, as long as all stakeholders are aware of the components and their functionality. The wireframes will be used to identify and name the components when assigning probabilities of use to the interface.

A second prerequisite is that the application developer have one or more clearly defined user personas or use case scenarios developed for the intended application. This use case should be crisply defined and include the affinities of the users expected to follow this use. Once both the use cases and wireframes have been developed, the process of developing the formal user model can begin. During this phase, each application component or feature is considered and a probability distribution describing the probability that the intended users will utilize this component is proposed. Additionally, some real-valued distributions can be assigned as well, such as the width of a panel, if the proposed interface allows interactive resizing. Whether or not a component is used by a hypothetical user following a use case scenario can be modeled as a Bernoulli distribution with probability of success α. Thus, the probability of use of a given component is considered a discrete random variable. In this way, variation between users can be simulated while enforcing patterns between expected groups.

Real valued components can be modeled using a continuous distribution. However, few values evaluated on an interface can be infinite, as most user interaction is constrained in some way. For example, panel widths cannot exceed 100% of the page width or be less than 0% of the width. Therefore, two potentially useful distributions are the gamma distribution and the binomial distribution. The gamma distribution is truncated at zero, with support from zero to infinity. The gamma distribution is effective for modeling positively real-valued phenomena, such as the zoom level of a map component. The binomial distribution is effective for modeling the dimensions of interface components. Given the need for mobile responsive applications, it is often effective the measure the dimensions of a component in percentage terms, rather than in pixels. This percentage can be modeled as a binomial distribution with 100 trials and a probability of success on each trial of α. Thus, as with the Bernoulli distribution described above, group-level similarities are imposed by group-to-group differences in α, while intragroup variation may remain, as it is likely to in real life. Once a distribution for each component of interest has been proposed by the model developer, it can be negotiated with stakeholders. While the application developer has a clear understanding of expected use, it may not align with the target user group's expectations. Because the distributions are, ultimately, a way of communicating between developers and stakeholders, efforts should be made to refine the distributions with the input of members of of the target use cases. Alternate distributions may be proposed, or, more likely, the distribution parameters may be tweaked to more accurately reflect the stakeholder's input. Stakeholders are likely to more effectively understand how that user group would actually utilize application components.

Communication and negotiation with the target users can be done in several ways. Informal interviews with key members of each stakeholder may be the best approach. Other social science data collection methods, including formal interviews or focus groups may also be effective. While an online survey would capture the opinion of more potential users, distribution negotiation should, in most cases, be limited to the users most qualified to comment on the potential

use cases of the user group. Real world uses may be compared and used to refine the distributions using Bayesian data assimilation once the application has been released.

Once the distributions of use have been proposed and negotiated, a user model is then available for subsequent analysis. This user model is used to generate a probability generated configuration (PGC) by drawing from the distributions of each component. To create a PGC, a single draw from each component's distribution is taken. Each draw is taken independently of all others. Therefore, a PGC is a configuration of the application potential under the given use case scenario. Variability is inherent within users within a given use case scenario. Because each draw is from a random distribution, this variation is captured while maintaining the salient features of scenario membership. Analyzing the set of PGCs, particularly if PGCs are generated from multiple scenarios, can yield insight into the variability between these scenarios.

Below, a case study describing the proposal and negotiation of the distributions for a new web-based interactive application is presented. PGCs are generated and subsequently analyzed and visualized to understand differences between user groups. Finally, initial data from beta testing is introduced and used to refine probability estimates using Bayesian data assimilation.

Aragon, C. R., Bailey, S. J., Poon, S., Runge, K., & Thomas, R. C. (2008). Sunfall: a collaborative visual analytics system for astrophysics. *Journal of Physics: Conference Series*, *125*(1), 012091. https://doi.org/10.1088/1742-6596/125/1/012091

Carroll, J. M. (2000). Five reasons for scenario-based design. *Interacting with Computers*, *13*(1), 43–60. https://doi.org/10.1016/S0953-5438(00)00023-0

Carroll, J. M., Rosson, M. B., Chin, G., & Koenemann, J. (1998). Requirements development in scenario-based design. *IEEE Transactions on Software Engineering*, *24*(12), 1156–1170. https://doi.org/10.1109/32.738344

Conati, C., Gertner, A., & Vanlehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted ...*. Retrieved from http://www.springerlink.com/index/uh10863178301011.pdf

Cunha, B., Pereira, J. P., Gomes, S., Pereira, I., & Santos, J. M. (2014). *Using personas for supporting user modeling on scheduling systems*. Proceedings of IEEE HIS. Retrieved from http://scholar.google.com/scholar?q=related:qmToQtxPzbsJ:scholar.google.com/&hl=en&as_sdt=0,5

G Smith Linden B.; York, J. (n.d.). Amazon.com Recommendations: Item-to-item Collaborative Filtering. Retrieved from http://books.google.com/books?id=axtzNQAACAAJ&dq=intitle:Amazon+com+Recommendations&hl=&cd=1&source=gbs_api

Gil, Y., Greaves, M., Hendler, J., & Hirsh, H. (2014). Amplify scientific discovery with artificial intelligence. *Science*, *346*(6206), 171–172. https://doi.org/10.1126/

science.1259439

Go, K., & Carroll, J. M. (2004). The blind men and the elephant: Views of scenario-based system design. *Interactions*, *11*(6), 44–53. https://doi.org/10.1145/1029036.1029037

Isbell, C. L., & Pierce, J. S. (2005). An IP continuum for adaptive interface design. *Proc of HCI International*. Retrieved from http://www.cc.gatech.edu/~isbell/papers/IPContinuum-IsbellPierce.pdf

Kazman, R., Abowd, G., Bass, L., & Clements, P. (1996). Scenario-based analysis of software architecture. *IEEE Software*, *13*(6), 47–55. https://doi.org/10.1109/52.542294{\&}url_ctx_fmt=info:ofi/fmt:kev:mtx:ctx{\&}rft_val_fmt=info:ofi/fmt:kev
based

Rosson, M. B. (2002). Scenario-Based Design. In J. Jacko & A. Sears (Eds.), *The human-computer interaction handbook fundamentals, evolving technology and emerging applications* (pp. 1032–1050). CRC Press. https://doi.org/10.1201/b11963-56

Rosson, M. B., & Carroll, J. M. (2005). *Scenario-Based Design*. Retrieved from http://s3.amazonaws.com/academia.edu.documents/30774225/Rosson_2002_sbd.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1489451535&Signature=G6I8G2Na61edTtXKTxfdC41mpa0%3D&response-content-disposition=inline%3B%20filename%3DScenario_based_design.pdf

Shneiderman, B. (2010). *Designing the user interface: Strategies for effective human-computer interaction*. Pearson Education India.

Zhang, C., Govindaraju, V., Borchardt, J., Foltz, T., Ré, C., & Peters, S. (2013). *GeoDeepDive: statistical inference using familiar data-processing languages*. New York, New York, USA: ACM. https://doi.org/10.1145/2463676.2463680