An IP Continuum for Adaptive Interface Design

Charles L Isbell and Jeffrey S Pierce

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{isbell, jpierce}@cc.gatech.edu

Abstract

The promise of adaptive user interfaces is that they will provide a flexible mechanism for systems to adapt to the needs of different users for a variety of tasks. As we consider such systems a basic trade-off arises between the responsibilities of the system and the user. We present a continuum that expresses potential balances of proactivity between the user and the system: what combinations of actions by those two could be responsible for accomplishing a particular task. In addition to describing the continuum, we describe how a set of example applications fit into it and discuss its implications. In particular, our continuum provides a framework and vocabulary for discussing and comparing adaptive interfaces. The continuum also provides directions for future work by suggesting potential interfaces and identifying new research directions, such as designing interfaces to maximize effective feedback.

1 Introduction

A primary strength of computational devices is their ability to automate tasks. The increasing computational power of our devices provides us with an opportunity to help users complete tasks more efficiently by shifting some of the burden for accomplishing those tasks from the user to the system. In particular, we can apply the available computational power to determine how and when to proactively assist users.

If we focus on the idea of assistants in the broad sense, we see that such systems would need to be aware of and able to adapt to the particular needs of specific users. After all, individual humans are different. More to the point, even a single person's habits and desires change over time. As such, creating a true personal assistant will require access to the user's personal data and individual behavior and will result in moving beyond fairly static personal computers to provide an adaptive and personalized information environment.

There are trade-offs in shifting the balance between user and system proactivity. Users (in general) know what tasks they want to accomplish and have the requisite information, while computers can only infer what users want and must typically work with limited information. On the other hand, users have limited attention, memory, and cognitive capabilities, while computers have large storage, multi-tasking, and computational capabilities. When designing an adaptive interface, we must find an appropriate balance for those (and other) trade-offs; however, finding that balance is difficult because we lack an understanding and appreciation of the full extent of the interface design space.

Consider the problem of designing a next-generation alarm clock. Users are currently fully responsible for deciding on their alarm time. While users could in principle optimize that decision by explicitly gathering information about their schedule, the weather, and traffic, they instead rely on routines because their routines are usually sufficient. A next-generation alarm clock could help users deal with situations where their routines fail because of, for example, an unusual or cancelled meeting. When designing such a clock, we might turn to previous research on adaptive interfaces for inspiration. That research suggests three common approaches. First, the alarm clock could suggest a possible time when the user initiates an alarm time change. Second, the alarm clock could initiate a dialog with the user to determine whether to change the alarm time. Third, the clock could automatically change the alarm time for the user.

While successful for other domains, these approaches are problematic for an alarm clock. The first approach can fail because users, relying on routines, may not realize that they need to change their alarm time. The second approach can fail because dialog requires a fair amount of the user's attention and the clock is likely to err on the side of caution

(asking often) because it lacks complete information. As a result, its false positive rate is likely to be high, distracting the user unnecessarily. The third approach is also likely to fail because the clock lacks information, for example that the user would prefer to use the time from a cancelled meeting to work on a paper rather than sleep in.

The more general problem is that previous research provides point designs for adaptive interfaces, but has yet to significantly map out the design space and describe the characteristics of different regions within that space. As a step toward that goal, we introduce an interface-proactivity (IP) continuum. The continuum provides a framework and vocabulary for discussing and comparing adaptive interfaces. The continuum also provides directions for future work by suggesting potential interfaces and identifying new research directions. In the remainder of this paper we describe the IP continuum, describe how a set of example applications fit into it, and discuss its implications.

2 The IP Continuum

The IP continuum expresses possible balances of proactivity between the user and the system: what combination of actions by those two could be responsible for accomplishing a particular task. On the far left of the continuum, the user is completely and solely responsible for performing any and all actions. This point on the continuum is the province of human-computer interaction (HCI) researchers, who explore how to create direct manipulation and other traditional graphical users interfaces in order to allow users to express themselves effectively. On the far right, the system is completely and solely responsible for acting. This point on the continuum is the province of artificial intelligence (AI) researchers, who explore how to enable computers to make the best decisions possible with available information. So-called "intelligent user interfaces" (including adaptive interfaces) that draw on both HCI and AI to a greater or lesser degree occupy the space in the continuum between those two end-points.

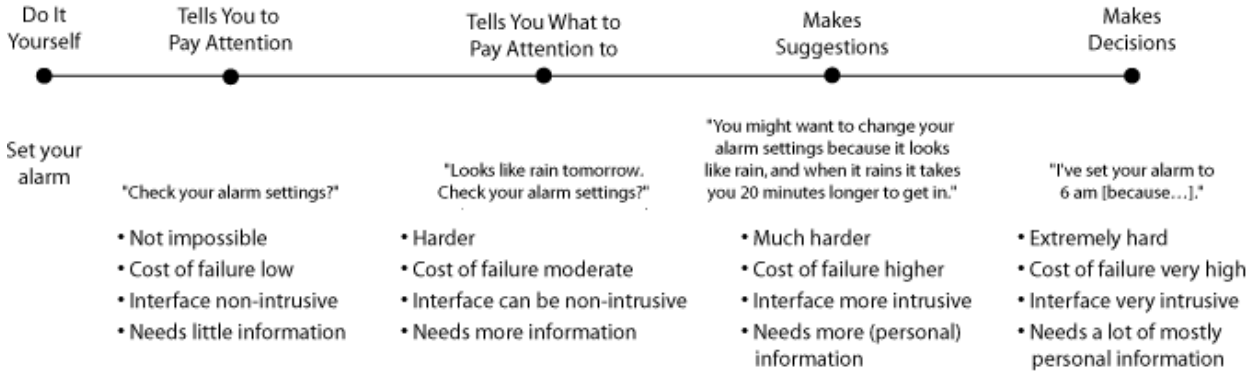

Figure 1: Potential Next-Generation Alarm Clock Interfaces Distributed Along the IP Continuum

Figure 1 shows potential interfaces for a next-generation alarm clock and some of their characteristics distributed along the IP continuum. On the far left we have the point occupied by today's clocks, where the user is solely responsible for setting the alarm. On the far right we have a (possibly) ideal alarm clock that automatically determines the best possible alarm time and sets it. Although there are many possible interfaces between those two points, we highlight three. The clock could assume a little responsibility by drawing the user's attention to the clock in general. That helps the user determine when they might need to change their routine, but does not narrow down for the user what information should inform that decision. The clock could go a step further and draw the user's attention to a particular piece of information, but still leave the decision completely up to the user. That step reduces the information the user must consider, but introduces the risk that the clock could accidentally draw the user's attention to the wrong information. The clock could assume even more responsibility and make a specific suggestion to the user. The user then must just decide whether or not to accept the suggestion, but without an understanding of the system's reasoning the user could decide incorrectly. We note that the continuum shows the relative, rather than absolute, positions of the described interfaces; determining the absolute positions of these and other interfaces is an important future step. Even without that step, determining the relative amount of responsibility assigned to the user and the system is useful for characterizing existing interfaces and suggesting new ones.

Although we have described interfaces as occupying a single point along the continuum, an application's interfaces may also occupy a section of the continuum. In those applications, the particular interface (and point along the continuum) that an application chooses can depend on a variety of factors, including the user's preferences, the current usage context, and the interface's confidence in its prediction of the appropriate next action (with more confidence leading to a higher degree of proactivity). We describe the section of the IP continuum occupied by an application as lying between the least and most proactive user interfaces that it can present to users. This description typically results in a continuous section of the continuum with gaps to the left and right.

2.1 Associated Metrics

In addition to the level of interface proactivity, there are other, associated metrics we can consider. While each of these metrics is not completely dependent on the level of interface proactivity, they are also not completely independent of it. As a result, choosing a point along the continuum for an interface will affect more than its proactivity. Alternately, if one or more of the associated metrics are the primary constraints on the interface, then they will also constrain the interface's possible positions along the continuum.

The level of system proactivity directly impacts the *benefit to the user* when the system is correct and the *cost to the user* when the system makes a mistake. At the left extreme of the continuum, the user bears no cost but also realizes no benefit. At the right extreme, the user may derive significant benefit when the system is correct, but the cost of failure may be so high that the user is unwilling to use or trust the system. Between those two extremes, the benefit of success and cost of failure rise as interfaces move from left to right. The benefit of success and cost of failure also depend on the application domain and particular situation: an application that assigns the incorrect time to a meeting will likely cost the user less than an application that automatically deletes any email that it thinks is spam, unless the missed meeting was with the user's boss and results in the user getting fired. Other metrics affecting the benefits and costs include the attention requirement and the likelihood of success. We discuss those metrics in more detail below.

Users' attention is a scarce resource. The *attention requirement*, or how much of the user's attention an interface consumes, can impact its benefit when it is right and its cost when it is wrong. In general the attention requirement for an interface drops as the interface moves from left to right along the continuum: an interface that the user controls directly consumes all of his attention, while an interface that does the right thing for the user without requiring his attention increases the benefit it provides by allowing the user to focus on other things. On the other hand, an interface that does the wrong thing without the user realizing it can make the cost of failure even worse because the user may discover the mistake too late to fix it or may not discover it at all. Interfaces do have some freedom to vary their attention requirement independently of their proactivity. A highly proactive interface could, for example, require more of the user's attention. An interface that did the right thing but kept notifying the user about its actions might be too annoying to use, while an interface that occasionally did the wrong thing but provided enough notification for the user to notice and correct it might provide significant value.

An adaptive interface that provides a small benefit to the user when it acts correctly but imposes a large cost when it is wrong might still be worth using if the *likelihood of success* is high enough. As a simple example, most users would probably be willing to use an interface that was statistically likely to save them five minutes a day, even if it was also likely to cost them a day to fix an error every two years. In general the likelihood of success decreases as interfaces move from left to right along the continuum.

The willingness of users to employ a particular interface also depends on its *frequency of action*. Users might be willing to employ an interface that is right only 75% of the time if it only acts once a year. By contrast, users would probably avoid an interface with the same likelihood of success if it acted every minute, because on average it would make a mistake every four minutes. Although the frequency of action tends to depend heavily on the application domain, the level of proactivity can also affect it. In particular, interfaces tend to act less frequently as they move from left to right because of the increased risk of making a mistake.

In order take the right course of action, an application needs data, and the more proactive an application is the larger the *amount and quality of data* it needs. Given access to a user's schedule, our hypothetical next-generation alarm

clock could easily draw the user's attention to unusual early morning meetings and let the user decide whether or not to change the alarm time. If we want the alarm clock to also be able to accurately suggest an alarm time, we need to provide it with a model of how long it takes the user to travel at different times of the morning, as well as information about when the user is likely to shift their alarm time instead of just spending less time getting ready. If we want the clock to proactively change the alarm time without consulting or notifying the user, we need even more information: what other morning events might not be in the user's calendar (e.g. planning to run errands), likely morning traffic and weather conditions, how much sleep the user got the night before, etc.

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. We note that this increased need for feedback can conflict with the decreased attention requirement as interfaces move to the right. Adaptive interfaces may gather implicit rather than explicit feedback to lower their attention requirement, but doing so can reduce the quality of the feedback.

The required amount of data imposes two costs: the cost of gathering it and the cost of storing it. The *cost of gathering data* may be negligible or it may be prohibitively high. In general the cost of gathering data increases as interfaces become more proactive. For example, drawing the user's attention to an unusual meeting on their calendar only requires access to information that is often already available digitally, but suggesting a new alarm time could require determining how long it takes users to get ready for work and to commute. Gathering that data could require extensive instrumentation of their home, car, and office. Applications can make do with less or lower quality information, but they typically pay with a lower likelihood of success.

The cost of storing the data is one of the *resource costs* that an application imposes on the computer running it. Applications that require a lot of information will often need to store it for some period of time in order to build up a model of the user's activities. The more information required and the longer it is needed, the more storage the application will need. The other primary resource cost of adaptive interfaces is computation. More proactive applications are likely to be more computationally difficult, increasing the resource demand on the host computer.

2.2 Related Work

The IP continuum extends previous work that explicitly or implicitly describe the design space for intelligent interfaces. Buxton divided interfaces based on whether they supported foreground or background interaction (Buxton, 1995). Those two categories match the leftmost and rightmost points on our continuum respectively. Hinckley at al extended Buxton's work by arguing that interfaces really occupied more of a continuum between foreground and background (Hinkley, et al, 2005), although they did not characterize that continuum in more detail.

Research on intelligent agents divides that middle ground into advisors and assistants (Liebeman and Selker, 2003). *Advisors* suggest courses of action while *assistants* act on behalf of the user. While those categories are a useful step toward characterizing interfaces along the continuum, they do not suggest how designers should compare the relative merits of an interface within a category. For example, knowing that the clock interface that draws the user's attention to a cancelled meeting and an interface that suggests changing the alarm time an hour later are both advisors does not help a designer determine which is better suited to his application.

Mixed-initiative interfaces extend the previous categorizations by proposing that interfaces should have mechanisms, such as dialog, that allow the user and agent to collaborate to refine a potential action. Horvitz's principles of mixed-initiative user interfaces (Horvitz, 1999) argue that interfaces should provide a variety of possible actions, explicitly considering the expected value of each in light of the potential costs, benefits, and uncertainties, and gracefully degrading the level of service (which we would term proactiveness) when faced with increasing uncertainty. We extend Horvitz's work by providing a concrete and continuous mechanism, the IP continuum, for describing and relating the possible actions that an interface could take and by discussing additional associated metrics.

While not explicitly attempting to characterize interfaces along a continuum, Bellotti et al (2002) posed a series of questions intended for sensing systems whose answers are also useful for characterizing aspects of adaptive systems.

Action and *address* are useful for describing both how the user requests that the system perform a particular action and how the user provides feedback about an action. *Alignment*, or how the user knows that the system is doing the right thing, is directly related to an interface's attention requirement. Finally, *accident* describes how an interface allows users to avoid mistakes, which relates to the attention requirement, likelihood of success, and cost of failure.

3 Fitting Applications Into the Continuum

In this section we describe how our work both fits into and is inspired by the IP continuum. We also describe how a small selection of adaptive interfaces developed by other researchers fit into the IP continuum to demonstrate that the continuum can describe work beyond our own.

3.1 Our Applications

Next-Generation Alarm Clock. Returning to our earlier example, we have designed a next-generation alarm clock. The choice of an alarm clock and its design was the result of a study of 19 participants that we conducted on the kinds of information that users needed when making a number of decisions during their night-time and morning routines (Landry, Pierce & Isbell, 2004). In addition to providing the ability to decide when to wake up and when to get out of bed, alarm clocks are typically nearby when users make the two other decisions we identified from our study: what to wear, and when to actually leave. Extending the traditional alarm clock to provide support for those decisions was thus a natural choice.

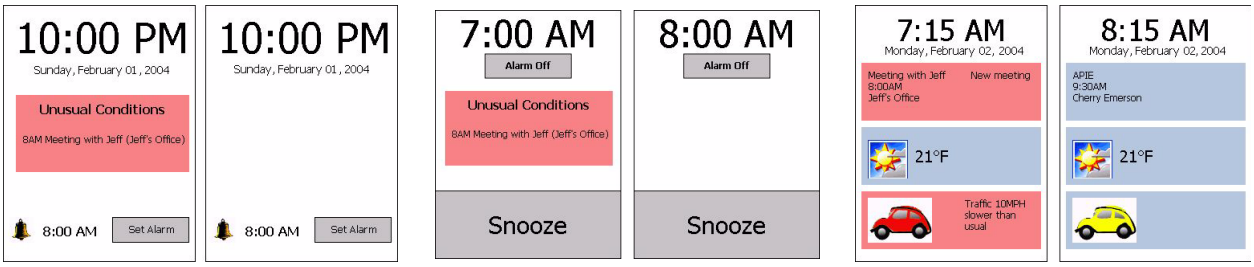

Figure 2: Examples of (left) the Before Bed, (middle) During Alarm, and (right) After Alarm interfaces when the system has and has not detected an unusual schedule event or forecast weather condition.

Although the information participants found useful for making decisions varied, there was a small set of commonly used types. All participants said that their schedule affects when they leave home, because of either an early first appointment or a desire to arrive at work at a particular time. Some also varied their routine based on weather and traffic. That information is typically available online at relatively low computational cost, but current approaches do not make accessing that information sufficiently easy for the user.

As noted earlier, a clock could assume a little or a lot of responsibility. Some researchers have suggested creating an intelligent alarm clock that takes a great deal of responsibility and changes the user's alarm time, but we do not believe that current technology can sufficiently reduce the risk of false positives and false negatives. On the other hand, the task is not impossible. As a practical matter, we believe that in the majority of cases users will not need to revise their normal routine (e.g. their alarm time, how warmly they dress, when they leave for work). Therefore, determining whether or not a revision is necessary should not place a huge burden on the user. Users need to be able to assess at a glance whether or not they need to modify their routine. We therefore designed the user experience to highlight unusual situations, allowing users to inquire further for additional information if it appears necessary.

Figure 2 shows interfaces for our next-generation alarm clock. There are three states: before the user has gone to bed, while the alarm is activated, and after the user has turned off the alarm (and stopped snoozing). In the first two cases, the default interfaces (on the right) look like a normal alarm clock. When unusual events have been detected, the interface displays it prominently, highlighting in red the relevant condition and displaying a brief indication of what

the system believes is unusual about it. A casual glance then lets the user know whether to pay attention. Because users do not always look at the display before hitting the snooze button, the tone of the alarm changes when the system has detected an unusual condition that may necessitate a change in routine. Once the user has turned off the alarm, the interface displays the user's first appointment, the current weather conditions, and the current traffic conditions. At any time our system allows the user to ask for more detail about displayed information.

This particular domain has several properties typical of domains where adaptive interfaces might be useful. The decision process is typically Markovian or memoryless, i.e., one need only know the current state of the world (e.g., the weather for today as opposed to the weather from last week) to make a decision; the information a user needs to make a decision is accessible but requires some effort; and the decision process has low entropy, i.e., the user has a routine that is almost always sufficient. As a result, the benefit of any single interaction will usually be zero, and the user may feel that it is not worthwhile to expend any great effort to make an optimal decision. So, rather than the user having to benefit from every interaction, users will instead benefit from awareness of occasional events that provide discrete but tangible benefits (e.g., not missing an important meeting). Those discrete benefits will accumulate over years and decades, eventually yielding a significant overall quality-of-life improvement.

The calendar's consumption of resources is quite low. Our prototype used a PDA connected via Bluetooth to an internet-connected home personal computer (PC). The home PC collected the weather, traffic, and schedule data using a variety of internet and web services. Again, because of the memoryless nature, the amount of data that had to be collected and maintained was minimal, depending only upon the current weather and traffic and the user's calendar.

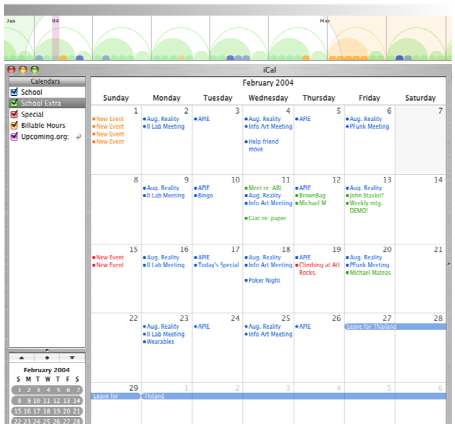

Figure 3: The arcs provide a radar-like overview of events that deviate from a regular routine.

Next-Generation Calendar. All the participants in the alarm clock study indicated that their morning schedule was a primary driver for when they wanted to leave home. Again, for many users their schedule is relatively fixed, at least with respect to when they first need to be at work or school; however, calendar information is often composed not only of a large number of routine events but also a small number of unusual events. Calendar arcs is a novel visualization for calendar data that leverages the regular nature of most scheduled events. As with the interface for the alarm clock, calendar arcs highlight deviations from a user's routine. Figure 3 shows calendar arcs above a traditional calender view.

We provide users with information about the "unusualness" of upcoming calendar events by detecting long-term regular statistical patterns. In particular, we use independent components analysis (Bell & Sejnowski, 1995) to extract the recurring "causes" of data, allowing automatic and principled detection of routines and deviations from routines. Calendar arcs then make the patterns and deviations obvious to the user. This is the primary computational cost for the system; otherwise, the difficulty of acquiring the data and the cost of storing it is only incrementally more than that faced by any calendar program. Beyond that, this application shares a great deal in common with the alarm clock. Aside from being able to use the same mechanism for learning routine schedules, it occupies much of the same space on the continuum. This is driven by the kind of question that the applications help the user to answer. In both cases the question is not *what do I have to do?*, but rather *is there anything keeping me from following my normal routine?*

The Next-Generation Universal Remote. Another domain that rests on strong routines is that of building interfaces for interacting with household appliances. Because of limited screen real estate and the plethora of devices and commands available to the user, a central problem for ubiquitous computing is predicting which appliances and devices the user wishes to use next so that interfaces for those devices can be made available. We have developed a personalized adaptive remote (Isbell, Omojokun & Pierce, 2004) that automatically discovers the user's common tasks (e.g., watching a movie, or surfing TV channels), predicts the task that the user wishes to engage in, and generates an appropriate interface that spans multiple devices. We were able to show that it is possible to discover and cluster collections of commands that represent routine tasks and to use immediate history to predict the next task reliably.

An interface that lived on the very far right of the continuum would simply take the next action for the user (e.g., turn off the lights, turn off the phone ringer, and start the DVD player rather than give the user the interface for doing such things), but that would be extremely intrusive and unacceptably annoying whenever it was wrong (and possibly almost as annoying when it was right). Moving much farther to the left would yield a system that would prompt the user whenever it believes that it has discovered a new task and give the user the opportunity to accept or edit the new interface. This would allow the user more control but would also require a great deal more attention. Our decision was to build a system that lies in between. Whenever the system infers a new task and associated interface, it creates an interface for it, adding that to its repertoire. Whenever it then believes the user is about to be enter into a new task it provides the interface for that task. Because it is possible to predict tasks (as opposed to actions) reliably, this approach seems to be the best for providing benefit with a high chance of success. Although this choice does not put the interface as far right as possible, it is further to the right than the systems we have described so far.

SWIMM. SWIMM is a very different application. It builds models of an individual's music listening habits and plays appropriate music for him. Our user studies of music listening revealed two things. First, users are not very good at describing the sort of music they want to listen to (although they can provide examples). Second, users are at least as interested (if not moreso) in having personalized radio stations as they are in creating short playlists. From a modeling point of view this raises a number of interesting mathematical as well as user interface questions. For our purposes here, this domain offers an interesting counterpoint to those above. Although there is a notion of usual behavior in this problem, it is somewhat different than the routines we have described so far.

Our interface sits in a different space in the continuum, making decisions but allowing the user to modify them. After the initial set up, the application sits in the background playing music while displaying its idea of acceptable next songs. When it gets it right, the user should hardly notice that it is around. When the system has made a mistake or when the user wants to engage in steering or teaching the application, it is a simple matter for the user to edit the upcoming selection of songs. The user's interaction is simple. For example, in creating a new radio station, the user selects a few sample songs, creating an initial distribution over songs. SWIMM continually updates the ordered sequence of songs to be played on the radio station, displaying a list of upcoming songs. As the user listens to, removes, adds, locks-in, or reorders songs on the list, SWIMM changes its model of both appropriate songs for that station and relationships that describe their relative ordering (e.g., SWIMM might discover that the user prefers that songs that follow one another have similar tempo). In short, the system begins near the left side of the continuum but moves to the right as quickly as it can.

The resource costs are significant. Further, the kind of information needed to build a good model of the user is not easy to extract, highlighting the need for powerful techniques for soliciting implicit feedback. On the other hand, the cost of failure is only severe if the failure is egregious. For example, while there are unacceptable sequences of songs, there are many satisfactory ones—even if they are in some sense suboptimal. In other words, there is not a correct answer so much as good ones, so the likelihood of "success" could be quite good.

Attachments. The ability to add attachments to email is both useful and popular, but existing mechanisms have a weakness: users can and do forget to attach desired files. In an informal survey of 50 people who frequently send email, 49 admitted to occasionally forgetting to add an attachment to an email message. The tendency to forget attachments is a direct result of the fact that users add attachments infrequently; as a result, the attachment process is not part of the workflow associated with writing the majority of messages. In other words, attachments are not a part of a user's normal routine. Clearly mail composition applications should move to the right on the IP continuum.

Figure 4 shows a set of possible interfaces along the IP continuum, labeled A-F. Interface A reminds users by playing a simple tone, while B colors the background of the message body pale yellow. These two interfaces provide notification that the user needs to pay attention to something, but do not specify what. C provides a more specific notification by coloring the attachment line pale yellow. D goes one step farther by attempting to draw the user's attention to the attachment button in order to suggest how to rectify the problem. E and F explicitly query about adding an attachment when the user clicks Send or uses the send shortcut key.

We conducted a Wizard of Oz study where 19 participants wrote three email messages for each interface. Users preferred the more proactive interfaces. A was clearly the least favorite, D, E & F were clearly the most favorite, and B

& C were ranked in the middle. 11 participants commented that the timing of reminders was important. They tended to like interfaces, such as E and F, that reminded them after they finished composing a message. D provided a reminder during message composition, but participants did not always immediately notice, so it received a higher overall ranking.

This application is a case where there is a correct answer. The user wants to send an attachment or does not. One would think then that it is important to be as accurate as possible, reducing both false negatives and positives. In fact, it turns out that the consequences of error are more than simply annoying the user. Six different participants tried to attach files after erroneous notifications. They said that the system was insisting they attach a file, and they felt obligated to do so.

Unidad. We end this section with a problem much larger in scope. Our work is preliminary, but it is instructive to explore it here. The increasing ubiquity of devices enhances users' access to computation, but also sharply decreases the ease of managing data. In particular, users do not carry all of their devices with them, and therefore must frequently transfer files between devices so they have the information on the right device at the right time. There are two related problems that arise: the difficulty of predicting in advance which data will be needed on which device, and the challenge in synchronizing and managing many pieces of data across several heterogeneous devices. Unidad is a set of user-friendly applications and middleware that provide the illusion that a user's data is available on every device she owns whenever she needs it.

Where would Unidad sit on the continuum? First, the system should handle the normal management of data synchronization. This pushes the system to the right. On the other hand, the user will want to pull it to the left for unusual or unusually important circumstances. This is the opposite of the other applications that we have discussed. There, the user is responsible for the routine and the application is responsible for the unusual.

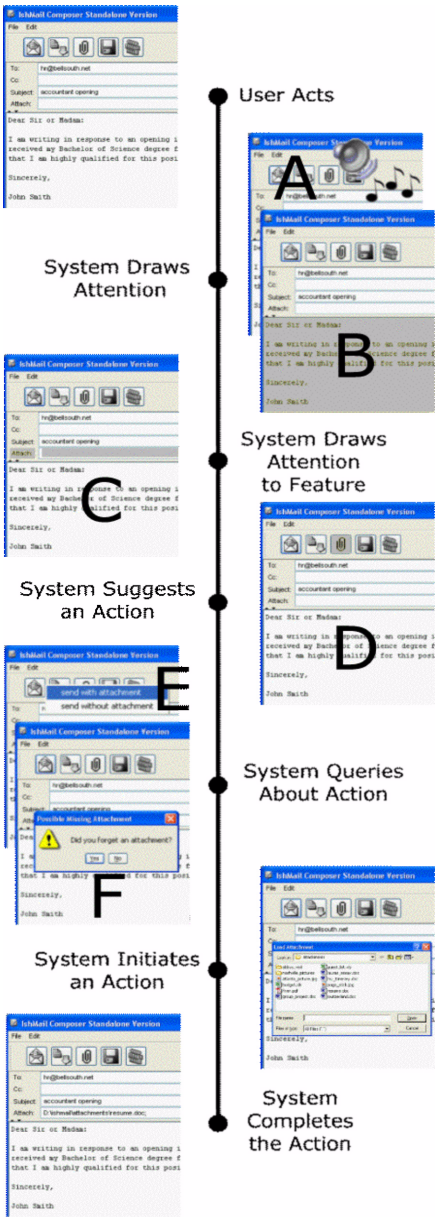

Figure 4: A set of interfaces for an attachment reminder

3.2 Other Applications

A common type of adaptive interface incorporates an "intelligent agent" that engages in discourse with the user to accomplish a particular task. The air travel assistant built with *COLLAGEN* (Rich & Sidner, 1998) is one example: the assistant engages the user in a conversation to help the user schedule a flight. Such an interface spans a section of the IP continuum because it can both suggest actions to the user and allow the user to adjust or parameterize its proposed actions. This assistant, and collaborative agents in general, occupy the middle of the continuum because they are designed to mimic the give-and-take of human conversation.

Lumiere (Horvitz, 1998) and *LookOut* (Horvitz, 1999) are examples of mixed-initiative applications. Lumiere is an automated assistant that can either help users find information or proactively offer assistance. When helping users find information, Lumiere takes a query for information from the user, combines it with sensed context about the user's activities and competencies, and returns a list of information topics that are appropriate for the tasks that it thinks the user is trying to accomplish. Lumiere can also proactively offer assistance by observing the user's actions. When it believes that the user is struggling to determine how to accomplish a particular task, it pops up a window displaying a list of help topics that are appropriate for the most likely tasks. Lumiere thus occupies a portion of the IP continuum that focuses on helping the user perform a task, rather than acting to perform the task for the user.

LookOut helps users place meetings arranged via email into their schedule. LookOut spans a wider section of the IP continuum by supporting three types of actions. In manual mode, LookOut pops up a dialog window when the user explicitly invokes it and fills it with inferred values. The user can adjust those values before saving the meeting in the calendar. LookOut's automated-assistance mode proactively pops up dialog windows with inferred values, while its social-agent mode tentatively adds meetings to the calendar before confirming them with the user. In addition to those specific points along the IP continuum, LookOut also moves presented interfaces around those points by varying their level of specificity based on its confidence in its predictions. For example, LookOut will suggest a particular day for a meeting if it cannot confidently determine a time, will suggest a particular week if it cannot determine a day, etc. LookOut thus spans a portion of the continuum ranging from assisting the user with parameter selection (on the left) to acting on the user's behalf and notifying the user of its actions (on the right).

4 Discussion and Future Work

The IP continuum allows us to describe the design space for adaptive interfaces and can help us identify new parts of the space to explore, but we also need to develop a more general understanding of the characteristics of the various regions of the design space. To develop that understanding we will need to draw on tools from both artificial intelligence and human-computer interaction.

In order to make our systems proactive, we need techniques that help the system infer when and how to be proactive. We have focused on machine learning for this purpose. Machine learning (ML) is that field of AI that concerns itself with building artifacts that adapt their behavior over time (Mitchell, 1997). Broadly speaking there are three kinds of machine learning: unsupervised, supervised, and reinforcement learning.

Unsupervised learning is the problem of finding compact descriptions of a set of data. For example, given a set of songs, one could cluster them into sets of similar songs, identifying, say, the three kinds of songs in a user's collection. Supervised learning is function approximation. An algorithm takes labeled training data of the form <x,y>, and tries to learn a function, f, such that y=f(x). For example, given a set of email messages labeled as normal and spam, one might build a classifier that determines whether a new message was spam. By contrast, reinforcement learning (Kaebling, Littman, & Moore, 1996) proposes particular mechanisms by which agents take actions and learn policies that maximize their long-term expected utility based upon feedback from the environment. Unlike supervised learning, the feedback tells the agent only whether it is doing well, as opposed to whether it is correct. For example, a system might change its model of a user's listening habits based upon whether the user reorders its suggested songs.

The last two techniques are naturally suited to the class of problems we have explored here. Unfortunately, it is often unrealistic to obtain regular feedback from a user. This is unfortunate because the quality of ML algorithms usually suffers under a paucity of data. Of course, we are not usually wanting for data so much as data that is labeled. The importance of providing regular and useful feedback to machine learning algorithms suggests a new emphasis for the interface design process. The HCI community has traditionally stressed the importance of considering effectiveness and efficiency when creating interfaces. As we shift to creating adaptive interfaces, we will need to explicitly consider how to design interfaces to provide explicit and implicit (in particular) feedback without negatively impacting their effectiveness. We believe that research on designing for feedback is necessary.

We will also need to consider how to adapt traditional HCI evaluation tools. Formative studies, which concentrate on measuring users' reactions to interfaces and identifying where users encounter problems, should map fairly well from traditional to adaptive interfaces. Conducting summative studies, which typically emphasize quantitative measures of performance, will be more problematic. Because adaptive interfaces will work differently for every user, conducting rigorous evaluations across users will be difficult, if not impossible. In addition, adaptive interfaces will change over time, so that conducting an experimental study where users interact with an application for an hour will only tell part of the story. We will need to develop experimental tools that allow us to measure the impact of adaptive interfaces across a wide variety of users for a sustained period of time.

Finally, even characterizing the different regions of the design space is only a step along the path toward allowing developers to confidently incorporate adaptive interfaces into commercial applications. For that to occur, we must

turn the design of adaptive interfaces from an art to a science. Our goal must be to assemble a set of principles and a body of scientific knowledge (Shneiderman, 1995) that developers can draw on. Attaining that goal will require a significant amount of work, but we believe that describing the design space for adaptive interfaces using the IP continuum and beginning a more organized, principled exploration of it are important steps forward.

5 Conclusions

We present an IP continuum that expresses the possible balances of proactivity between the user and the system: what combination of actions by those two can be responsible for accomplishing a particular task. In addition to describing the continuum, we describe how a set of example applications fit into it and discuss its implications. Our continuum provides a framework and vocabulary for discussing and comparing adaptive interfaces. The continuum also provides directions for future work by suggesting new potential interfaces and identifying new research directions such as designing interfaces to maximize effective feedback.

6 References

Bell, A & Sejnowski, T. (1995). An Information-Maximization approach to blind source separation and blind deconvolution. Neural Computation, 7:1129-1159.

Bellotti, V., Back, M., Edwards, W. K., Grinter, R., Lopes, C., & Henderson, A. (2002). Making sense of sensing systems: Five questions for designers and researchers. Proc. ACM CHI 2002 Conference on Human Factors in Computing Systems, pp. 415-422.

Buxton, W. (1995). Integrating the Periphery and Context: A New Taxonomy of Telematics. Proceedings of Graphics Interface '95, pp. 239-246.

Hinckley, K., Pierce, J., Horvitz, E., and Sinclair, M. (2005). Foreground and Background Interaction with Sensor-enhanced Mobile Devices, to appear in ACM TOCHI Special Issue on Sensor-Based Interaction.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence.

Horvitz, E. (1999) Principles of mixed-initiative user interfaces. Proceedings of CHI 1999, pp.159-166.

Isbell, C., Omojokun, O., and Pierce, J. (2004) From Devices to Tasks: Automatic Task Prediction for Personalized Appliance Control. Personal and Ubiquitous Computing, vol. 8 nos. 3-4, pp. 146-153.

Kaelbling, L.P., Littman, M.L., and Moore, A.W. (1996). Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, 4:237-285.

Landry, B., Pierce, J., and Isbell, C. (2004). Supporting Routine Decision-Making with a Next-Generation Alarm Clock. Personal and Ubiquitous Computing, vol. 8 nos. 3-4, pp. 154-160.

Lieberman, H., and Selker, T. (2003). Agents for the User Interface. Handbook of Agent Technology. Jeffrey Bradshaw, ed. MIT Press.

Mitchell, T (1997). Machine Learning. McGraw Hill.

Rich, C. and Sidner, C. (1998). COLLAGEN: A Collaboration Manager for Software Interface Agents. User Modeling and User-Adapted Interaction, vol. 8 no. 3/4, pp. 315-350.

Shneiderman, B. (1995). Looking for the bright side of user interface agents. ACM interactions, 2(1), pp. 13-15.