EXHIBIT 29

Exhibit C-11

Claim Chart Applying CyberDesk Against the '843 Patent

CyberDesk was known and/or publicly used in the United States at least by 1997. It therefore constitutes prior art under at least pre-AIA 35 U.S.C. § 102(a). As shown below, CyberDesk anticipates and/or renders obvious claims 1, 8, 13, 15, 17, 18, 19, 23, and 30 of the '843 patent, at least as CyberDesk was known, used, and described in (1) Dey, Anind et al., *CyberDesk: A Framework for Providing Self-Integrating Ubiquitous Software Services*, Technical Report, GVU Center, Georgia Institute of Technology, GIT-GVU-97-10, June 1997 ("CyberDesk Technical Report"); (2) Dey, Anind et al., CyberDesk: A Framework for Providing Self-Integrating Ubiquitous Software Services, UIST 97, ACM 0-89791-881-9/97/10 ("CyberDesk Summary"); and (3) Wood, Andrew et al., *CyberDesk: Automated Integration of Desktop and Network Services*, CHI 97, Atlanta GA, Mar. 22-27, 1997, ACM 0-89791-802-9/97/03 ("CyberDesk Technical Note"). If the Judge or Jury finds that CyberDesk does not anticipate a particular claim, then CyberDesk still renders the claim obvious for the reasons discussed in Exhibit F.

'843 Patent Claims	**Disclosure**
Claim 1	
A computer-implemented method for finding data related to the contents of a document using a first computer program running on a computer, the method comprising:	CyberDesk discloses a computer-implemented method for finding data related to the contents of a document using a first computer program running on a computer. See, e.g., CyberDesk Technical Report at 1, col. 1: "Current software suites suffer from problems due to poor integration of their individual tools. They require the designer to think of all possible integrating behaviours and leave little flexibility to the user. In this paper, we discuss CyberDesk, a component software framework that automatically integrates desktop and network services, requiring no integrating decisions to be made by the tool designers and giving total control to the user." See, e.g., CyberDesk Technical Report at 1, col. 1: "In response, software companies have been adopting the notion of component software: using small software modules as building blocks for a larger application. While there are many competing standards (OLE [11], Active X [10], Java Beans [6], OpenDoc [1]), the prevailing view is to provide a framework which programmers and sophisticated users can build upon to create desired application suites." See, e.g., CyberDesk Technical Report at 1, col. 2: "In this paper, we present CyberDesk system, a component software framework that relieves most of the burden of integrating services from both the designer of individual services and the end user, provides greater flexibility to the user, and automatically suggests how independent services can be integrated in interesting ways."

Exhibit C-11

’843 Patent Claims	Disclosure
	See, e.g., CyberDesk Technical Report at 1, col. 2: “CyberDesk is a component-based framework written in Java, that supports automatic integration of desktop and network services [16]. The framework is flexible, and can be easily customized and extended.” See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): “The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend’s house but nobody is home. The system asks if he wants to check his friend’s calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author’s contact information in the contact manager, call the author, or send an e-mail to the author.” See, e.g., CyberDesk Summary at 75 (including fig. 1): “The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy’s phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood’s contact information in the contact manager (d).” See, e.g., CyberDesk Technical Report at 4, cols. 1-2: “All of the desktop applets currently being used in CyberDesk (2 e-mail browsers, contact manager, 2 calendar managers/schedulers, scratchpad) were previously written by other Georgia Tech students.” See, e.g., CyberDesk Technical Report at 5, col. 2—6, col. 1 (including fig. 3): “The example below is the wrapper for the Contact Manager (see Figure 3), and it extends the ContactApplet class (the original application class). . . . Lookup an entry for the name in the

Exhibit C-11

'843 Patent Claims	**Disclosure**
	ContactManager." See, e.g., CyberDesk Technical Report at 6, col. 1 (figure 3): "Screenshot of contact manager being used with CyberDesk. The user selects the string 'Andy Wood' in the –mail tool (a). CyberDesk offers some suggestions (b): search using AltaVista, look up a phone number using Switchboard (c), and look up the name in the contact manager (d)." See, e.g., CyberDesk Summary at 75 (including fig. 1): "The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy's phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood's contact information in the contact manager (d)."
displaying the document electronically using the first computer program;	CyberDesk discloses displaying the document electronically using the first computer program. See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): "The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend's house but nobody is home. The system asks if he wants to check his friend's calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author's contact information in the contact manager, call the author, or send an e-mail to the author." See, e.g., CyberDesk Summary at 75 (including fig. 1): "The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a

Exhibit C-11

'843 Patent Claims	Disclosure
	phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy's phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood's contact information in the contact manager (d)." See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): "The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate 'ActOn' window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight 'Gregory Abowd' causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn't have Gregory's phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f)." See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): "The CyberDesk system has a simple but innovative architecture. It is based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in the form of suggestions to the user, via the ActOn Button Bar. It is through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below."

Exhibit C-11

'843 Patent Claims	Disclosure
	See, e.g., CyberDesk Technical Report at 6, col. 1 (figure 3): "Screenshot of contact manager being used with CyberDesk. The user selects the string 'Andy Wood' in the –mail tool (a). CyberDesk offers some suggestions (b): search using AltaVista, look up a phone number using Switchboard (c), and look up the name in the contact manager (d)."
while the document is being displayed, analyzing, in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information;	CyberDesk discloses, while the document is being displayed, analyzing, in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information. See, e.g., CyberDesk Tehnical Note at 553, cols. 1-2: "Finally, the user's selection, plus any extra type information generated by the conversion process, is observed by the ActOn integrating applet and a set of potential actions for that data is suggested. For example, a name is accepted as input by the Switchboard service, and so ActOn creates a button that suggests that integrating behaviour. Clicking on the button invokes the Switchboard service, completing the integration without requiring any change to the functionality of either service and without an programming effort from the user." See, e.g., CyberDesk Technical Report at 4, col. 2: "Data typing is used extensively in the interface declarations of the event sources and sinks that applications provide. The property field that corresponds to each interface declares the datatype/event that a component is interested in or can provide. The CyberDesk system takes advantage of the Java type system to do the data typing." See, e.g., CyberDesk Technical Report at 4, col. 2: "Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the 'To:' or 'From:' field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the

Exhibit C-11

’843 Patent Claims	Disclosure
	system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture.” See, e.g., CyberDesk Technical Report at 5, col. 1: “So, when any component generates a String selection even, the StringToEmailAddress converter (and any other observers) are notified, and the converter attempts to conver the given String object to an EmailAddress object (while other converters attempt to convert the object to another CyberDesk type). In the above scenario, this conversion was done when the user selected the e-mail address. The system initially saw the selected data as a String but with this converter, it also saw it as an EmailAddress. This results in two related data selection events to arrive at the IntelliButton: one containing a string and one containing an EmailAddress. The IntelliButton will therefore seek integrating behaviour for both these types, allowing the user to access EmailAddress-relevant services where originally they wouldn’t have had the option.” See, e.g., CyberDesk Technical Report at 5, col. 1: “Currently the list of CyberDesk types include: Date PhoneNumber, Mailing Address, Name, URL, and EmailAddress. If any of the conversions can be made, then the converter generates a second, but related, selection event containing the newly typed data and sends it to observing entities.” See, e.g., CyberDesk Technical Report at 5, cols. 1-2: “The CyberDesk framework was designed to be easily extensible. Simple extensions to CyberDesk include adding additional types, type converters, desktop services and network services. The real advantages with CyberDesk can be seen with more complex extensions that include adapting the behaviour of CyberDesk to individual use and creating more interesting integrating behaviour.” See, e.g., CyberDesk Technical Report at 5, col. 2: “An example converter is the StringToEmailAddress converter, which is a subclass of the ConversionApplet class. The code for this component and all components described in the paper can be viewed at http://www.cc.gatech.edu/fce/cyberdesk/samples. This converter looks at traditional ways of writing an e-mail address, and tries to map selected data to one of these ways. If it is successful, it returns an EmailAddress object. The ConversionApplet object is responsible for handling the ties to the CyberDesk framework.” See, e.g., CyberDesk Technical Report at 6, col. 1 (figure 3): “Screenshot of contact manager being used with CyberDesk. The user

Exhibit C-11

’843 Patent Claims	Disclosure
	selects the string ‘Andy Wood’ in the –mail tool (a). CyberDesk offers some suggestions (b): search using AltaVista, look up a phone number using Switchboard (c), and look up the name in the contact manager (d).” See, e.g., CyberDesk Technical Report at 7, cols. 1-2: “The next task involved adding the services that recognized the appropriate types and created ‘ActOn’ buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above.” See, e.g., CyberDesk Technical Note at 553, col. 1 (including fig. 3): “When the user selects information displayed by one service, say some text from the e-mail message, the type converters try recursively to see if the data can be converted to other types used in the system (e.g., a name in Figure 3). In the case of plain text, this could be done by comparing the string to common formats for representing the various types; for names you might use title firstname lastname, and similar patterns can be used for dates, URLs, e-mail and mailing addresses.” See, e.g., CyberDesk Technical Report at 8, cols. 1-2: “CyberDesk contains some simple notions of context. It knows the application a user is working with and the data (both type and content) the user is interested in (via explicit selection with the mouse). But CyberDesk has shown the potential for supporting higher level context. For example, if an e-mail message contains information about a meeting, and the user selects the message content, a type converter could potentially convert the text to a Meeting object to be inserted in the user’s Calendar Manager. Of course, retrieving context from arbitrary text is a very difficult problem being investigated by the AI learning community. But the power of CyberDesk supports the ability to use this higher level context, if available.” See, e.g., CyberDesk Technical Report at 9, col. 2: “Related Work. Pandit and Kalbags Selection Recognition Agent [12] attempts to address the same issues as CyberDesk. Unlike CyberDesk, it uses a fixed datatype-action pair, allowing for only one possible action for each datatype recognized. . . . Apple Data Detectors [2] is another component architecture that supports automatic integration of tools. It works at the operating system level, using the selection mechanism that most Apple applications support. It allows the selection of a large area of text and recognizes all user-registered dtatypes in that selection. Users view suggested actions by pressing a modifier key and the mouse button. Like CyberDesk, it supports an arbitrary number of actions for

Exhibit C-11

'843 Patent Claims	**Disclosure**
	each datatype."
retrieving the first information;	CyberDesk discloses retrieving the first information. See, e.g., CyberDesk Summary at 75 (including fig. 1): "The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy's phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood's contact information in the contact manager (d)." See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): "The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate 'ActOn' window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight 'Gregory Abowd' causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn't have Gregory's phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f)." See, e.g., CyberDesk Technical Report at 3, col. 2: "The previous three components discussed provide the core functionality of CyberDesk. Regardless of what tools the user wants to use, these three components are required. The fourth type of component, desktop and network services, are the actual services the user wants to access. Desktop services include e-mail browsers, contact managers, and schedulers. Network services include web search engines, telephone directories, and map retrieval tools. . . . One of the services available in CyberDesk is a gateway to the AltaVista search engine available on the web." See, e.g., CyberDesk Technical Report at 4, col. 1: "The wrapper must

Exhibit C-11

'843 Patent Claims	Disclosure
	have 'hooks' into the original application code to intercept and broadcast the appropriate data selection events (for the 'select' interfaces), and to execute a service on data passed to it (for the 'method' interfaces). At the time of developments, there were three ways to approach this problem for the 'select' interface. First, we could modify the original application's event processing loop to broadcast events in the CyberDesk fashion. Second, we could modify the original application code to make calls to a notification routine in the wrapper when data is selected. Third, we could rely on the original application have a suitable API for retrieving those events." See, e.g., CyberDesk Technical Report at 7, col. 1 (including fig. 4): "There were two reasons for integrating CyberDesk with LlamaShare. First, we wanted to illustrate the platform-neutrality and language-neutrality of the LlamaServer, which CyberDesk allows us to do. More importantly, however, CyberDesk's vision of ubiquitous information access was the deciding factor. While LlamaShare provides a concrete, visible object to represent the data on a mobile device, CyberDesk takes the approach that information is distributed throughout a rather nebulous information space (consisting of Internet, desktop, and mobile data) that can be retrieved at any moment depending on the context in which the user is currently working. This new metaphor of seamless integration between mobile data and Internet (remote) data was too good to pass up."
providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, the operation comprising (i) performing a search using at least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the search term in an information source external to the document, wherein the specific type or types of second information is dependent at least in part on the type or types of the first information, and (ii) performing an action using at least part of the second information;	CyberDesk discloses providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, the operation comprising (i) performing a search using at least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the search term in an information source external to the document, wherein the specific type or types of second information is dependent at least in part on the type or types of the first information, and (ii) performing an action using at least part of the second information. See, e.g., CyberDesk Technical Report at 1, col. 2: "The components in CyberDesk treat all data uniformly, regardless of whether the data came from a locally running application or from a service running on the World Wide Web (WWW). The services and applications themselves can be running anywhere, meeting CyberDesk's goal of providing ubiquitous access to services." See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): "The user highlights some text in the window of one service, and CyberDesk determines the tpe of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a

Exhibit C-11

’843 Patent Claims	Disclosure
	separate ‘ActOn’ window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight ‘Gregory Abowd’ causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn’t have Gregory’s phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f).” See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): “The CyberDesk system has a simple but innovative architecture. It is based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in the form of suggestions to the user, via the ActOn Button Bar. It is through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below.” See, e.g., CyberDesk Technical Report at 3, cols. 1-2: “The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as

Exhibit C-11

'843 Patent Claims	Disclosure
	parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them." See, e.g., CyberDesk Tehnical Note at 553, cols. 1-2: "Finally, the user's selection, plus any extra type information generated by the conversion process, is observed by the ActOn integrating applet and a set of potential actions for that data is suggested. For example, a name is accepted as input by the Switchboard service, and so ActOn creates a button that suggests that integrating behaviour. Clicking on the button invokes the Switchboard service, completing the integration without requiring any change to the functionality of either service and without an programming effort from the user." See, e.g., CyberDesk Technical Report at 3, col. 2: "The ActOn Button Bar, as described before, is simply the user interface for the integrating IntelliButton. We chose to keep the interface separate from the actual integrating functionality to all easier experimentation with alternative interfaces. Currently, the interface is very simplistic. It is a dynamically generated list of buttons, where each button corresponds to a particular service that can be executed based on an user-generated event and its corresponding data. The list of buttons is provided by the IntelliButton. . . . For example: Send e-mail to this EmailAddress using Netscape; Search for a string on the Web using AltaVista." See, e.g., CyberDesk Technical Report at 4, col. 1: "With this interface, this search would be suggested by the IntelliButton whenever a text string is the target of a selection (assuming the component in which the text selection is done, supports the 'select' interface)." See, e.g., CyberDesk Technical Report at 4, col. 2: "Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the 'To:' or 'From:' field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a

’843 Patent Claims	Disclosure
	String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture.” See, e.g., CyberDesk Technical Report at 5, col. 1: “So, when any component generates a String selection even, the StringToEmailAddress converter (and any other observers) are notified, and the converter attempts to conver the given String object to an EmailAddress object (while other converters attempt to convert the object to another CyberDesk type). In the above scenario, this conversion was done when the user selected the e-mail address. The system initially saw the selected data as a String but with this converter, it also saw it as an EmailAddress. This results in two related data selection events to arrive at the IntelliButton: one containing a string and one containing an EmailAddress. The IntelliButton will therefore seek integrating behaviour for both these types, allowing the user to access EmailAddress-relevant services where originally they wouldn’t have had the option.” See, e.g., CyberDesk Technical Report at 6, col. 2: “The following is an example network service: WhoWhereEmail. It is a gateway to the WhoWhere network service available on the web. When a name is input into the web service, a list of possible e-mail addresses corresponding to that name is returned. The CyberDesk network service takes a Name object, inputs it into the service and displays the results in a web browser.” See, e.g., CyberDesk Technical Report at 7, cols. 1-2: “The next task involved adding the services that recognized the appropriate types and created ‘ActOn’ buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above.” See, e.g., CyberDesk Technical Report at 9, col. 1: “Perhaps the biggest limitation of the system is the user interface implemented by the ActOn Button Bar. It consists of a window that displays a long list of suggested user actions. It is clear that the number of possible suggestions could quickly become overwhelming to the user. We are currently looking at different ways to adapt the interface to initially show actions that the user is likely to take, but provide a way for the user to see

Exhibit C-11

'843 Patent Claims	Disclosure
	other possible actions as well. We are also looking at different presentation methods for the suggestions, including pop-up hierarchical menus and document lenses [4]." See, e.g., CyberDesk Technical Report at 9, col. 2: "Related Work. Pandit and Kalbags Selection Recognition Agent [12] attempts to address the same issues as CyberDesk. Unlike CyberDesk, it uses a fixed datatype-action pair, allowing for only one possible action for each datatype recognized. . . . Apple Data Detectors [2] is another component architecture that supports automatic integration of tools. It works at the operating system level, using the selection mechanism that most Apple applications support. It allows the selection of a large area of text and recognizes all user-registered dtatypes in that selection. Users view suggested actions by pressing a modifier key and the mouse button. Like CyberDesk, it supports an arbitrary number of actions for each datatype." See, e.g., CyberDesk Technical Report at 1, col. 2: "The components in CyberDesk treat all data uniformly, regardless of whether the data came from a locally running application or from a service running on the World Wide Web (WWW). The services and applications themselves can be running anywhere, meeting CyberDesk's goal of providing ubiquitous access to services." See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): "The user highlights some text in the window of one service, and CyberDesk determines the tpe of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate 'ActOn' window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight 'Gregory Abowd' causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn't have Gregory's phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f)." See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): "The CyberDesk system has a simple but innovative architecture. It is

'843 Patent Claims	Disclosure
	based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in the form of suggestions to the user, via the ActOn Button Bar. It is through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below." See, e.g., CyberDesk Technical Report at 3, cols. 1-2: "The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them." See, e.g., CyberDesk Tehnical Note at 553, cols. 1-2: "Finally, the user's selection, plus any extra type information generated by the conversion process, is observed by the ActOn integrating applet and a set of potential actions for that data is suggested. For example, a name is accepted as input by the Switchboard service, and so ActOn creates a button that suggests that integrating behaviour. Clicking on the button invokes the Switchboard service, completing the integration without requiring any change to the functionality of either service and without an programming effort from the user." See, e.g., CyberDesk Technical Report at 3, col. 2: "The ActOn Button Bar, as described before, is simply the user interface for the integrating

Exhibit C-11

’843 Patent Claims	Disclosure
	IntelliButton. We chose to keep the interface separate from the actual integrating functionality to all easier experimentation with alternative interfaces. Currently, the interface is very simplistic. It is a dynamically generated list of buttons, where each button corresponds to a particular service that can be executed based on an user-generated event and its corresponding data. The list of buttons is provided by the IntelliButton. . . . For example: Send e-mail to this EmailAddress using Netscape; Search for a string on the Web using AltaVista.” See, e.g., CyberDesk Technical Report at 4, col. 1: “With this interface, this search would be suggested by the IntelliButton whenever a text string is the target of a selection (assuming the component in which the text selection is done, supports the ‘select’ interface).” See, e.g., CyberDesk Technical Report at 4, col. 2: “Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the ‘To:’ or ‘From:’ field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture.” See, e.g., CyberDesk Technical Report at 5, col. 1: “So, when any component generates a String selection even, the StringToEmailAddress converter (and any other observers) are notified, and the converter attempts to conver the given String object to an EmailAddress object (while other converters attempt to convert the object to another CyberDesk type). In the above scenario, this conversion was done when the user selected the e-mail address. The system initially saw the selected data as a String but with this converter, it also saw it as an EmailAddress. This results in two related data selection events to arrive at the IntelliButton: one containing a string and one containing an

Exhibit C-11

’843 Patent Claims	Disclosure
	EmailAddress. The IntelliButton will therefore seek integrating behaviour for both these types, allowing the user to access EmailAddress-relevant services where originally they wouldn’t have had the option.” See, e.g., CyberDesk Technical Report at 6, col. 2: “The following is an example network service: WhoWhereEmail. It is a gateway to the WhoWhere network service available on the web. When a name is input into the web service, a list of possible e-mail addresses corresponding to that name is returned. The CyberDesk network service takes a Name object, inputs it into the service and displays the results in a web browser.” See, e.g., CyberDesk Technical Report at 7, cols. 1-2: “The next task involved adding the services that recognized the appropriate types and created ‘ActOn’ buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above.” See, e.g., CyberDesk Technical Report at 9, col. 1: “Perhaps the biggest limitation of the system is the user interface implemented by the ActOn Button Bar. It consists of a window that displays a long list of suggested user actions. It is clear that the number of possible suggestions could quickly become overwhelming to the user. We are currently looking at different ways to adapt the interface to initially show actions that the user is likely to take, but provide a way for the user to see other possible actions as well. We are also looking at different presentation methods for the suggestions, including pop-up hierarchical menus and document lenses [4].” See, e.g., CyberDesk Technical Report at 9, col. 2: “Related Work. Pandit and Kalbags Selection Recognition Agent [12] attempts to address the same issues as CyberDesk. Unlike CyberDesk, it uses a fixed datatype-action pair, allowing for only one possible action for each datatype recognized. . . . Apple Data Detectors [2] is another component architecture that supports automatic integration of tools. It works at the operating system level, using the selection mechanism that most Apple applications support. It allows the selection of a large area of text and recognizes all user-registered dtatypes in that selection. Users view suggested actions by pressing a modifier key and the mouse button. Like CyberDesk, it supports an arbitrary number of actions for each datatype.”
in consequence of receipt by the	CyberDesk discloses, in consequence of receipt by the first computer

Exhibit C-11

'843 Patent Claims	**Disclosure**
first computer program of the user command from the input device, causing a search for the search term in the information source, using a second computer program, in order to find second information related to the search term; and	program of the user command from the input device, causing a search for the search term in the information source, using a second computer program, in order to find second information related to the search term. See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): "The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend's house but nobody is home. The system asks if he wants to check his friend's calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author's contact information in the contact manager, call the author, or send an e-mail to the author." See, e.g., CyberDesk Summary at 75 (including fig. 1): "The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy's phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood's contact information in the contact manager (d)." See, e.g., CyberDesk Technical Report at 1, col. 2: "The components in CyberDesk treat all data uniformly, regardless of whether the data came from a locally running application or from a service running on the World Wide Web (WWW). The services and applications themselves can be running anywhere, meeting CyberDesk's goal of providing ubiquitous access to services." See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): "The user highlights some text in the window of one service, and CyberDesk determines the tpe of the text to suggest how the user can invoke behavior in other services using that text. The

Exhibit C-11

’843 Patent Claims	Disclosure
	suggestions made by CyberDesk appear as a dynamic button bar in a separate ‘ActOn’ window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight ‘Gregory Abowd’ causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn’t have Gregory’s phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f).” See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): “The CyberDesk system has a simple but innovative architecture. It is based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in the form of suggestions to the user, via the ActOn Button Bar. It is through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below.” See, e.g., CyberDesk Technical Report at 3, cols. 1-2: “The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it

Exhibit C-11

’843 Patent Claims	Disclosure
	access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them.” See, e.g., CyberDesk Tehnical Note at 553, cols. 1-2: “Finally, the user’s selection, plus any extra type information generated by the conversion process, is observed by the ActOn integrating applet and a set of potential actions for that data is suggested. For example, a name is accepted as input by the Switchboard service, and so ActOn creates a button that suggests that integrating behaviour. Clicking on the button invokes the Switchboard service, completing the integration without requiring any change to the functionality of either service and without an programming effort from the user.” See, e.g., CyberDesk Technical Report at 3, col. 2: “The ActOn Button Bar, as described before, is simply the user interface for the integrating IntelliButton. We chose to keep the interface separate from the actual integrating functionality to all easier experimentation with alternative interfaces. Currently, the interface is very simplistic. It is a dynamically generated list of buttons, where each button corresponds to a particular service that can be executed based on an user-generated event and its corresponding data. The list of buttons is provided by the IntelliButton. . . . For example: Send e-mail to this EmailAddress using Netscape; Search for a string on the Web using AltaVista.” See, e.g., CyberDesk Technical Report at 4, col. 1: “With this interface, this search would be suggested by the IntelliButton whenever a text string is the target of a selection (assuming the component in which the text selection is done, supports the ‘select’ interface).” See, e.g., CyberDesk Technical Report at 4, col. 2: “Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the ‘To:’ or ‘From:’ field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress

Exhibit C-11

'843 Patent Claims	Disclosure
	object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture." See, e.g., CyberDesk Technical Report at 5, col. 1: "So, when any component generates a String selection even, the StringToEmailAddress converter (and any other observers) are notified, and the converter attempts to conver the given String object to an EmailAddress object (while other converters attempt to convert the object to another CyberDesk type). In the above scenario, this conversion was done when the user selected the e-mail address. The system initially saw the selected data as a String but with this converter, it also saw it as an EmailAddress. This results in two related data selection events to arrive at the IntelliButton: one containing a string and one containing an EmailAddress. The IntelliButton will therefore seek integrating behaviour for both these types, allowing the user to access EmailAddress-relevant services where originally they wouldn't have had the option." See, e.g., CyberDesk Technical Report at 6, col. 2: "The following is an example network service: WhoWhereEmail. It is a gateway to the WhoWhere network service available on the web. When a name is input into the web service, a list of possible e-mail addresses corresponding to that name is returned. The CyberDesk network service takes a Name object, inputs it into the service and displays the results in a web browser." See, e.g., CyberDesk Technical Report at 7, cols. 1-2: "The next task involved adding the services that recognized the appropriate types and created 'ActOn' buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above." See, e.g., CyberDesk Technical Report at 9, col. 1: "Perhaps the biggest limitation of the system is the user interface implemented by the ActOn Button Bar. It consists of a window that displays a long list of suggested user actions. It is clear that the number of possible suggestions could quickly become overwhelming to the user. We are currently looking at different ways to adapt the interface to initially show

'843 Patent Claims	Disclosure
	actions that the user is likely to take, but provide a way for the user to see other possible actions as well. We are also looking at different presentation methods for the suggestions, including pop-up hierarchical menus and document lenses [4]." See, e.g., CyberDesk Technical Report at 9, col. 2: "Related Work. Pandit and Kalbags Selection Recognition Agent [12] attempts to address the same issues as CyberDesk. Unlike CyberDesk, it uses a fixed datatype-action pair, allowing for only one possible action for each datatype recognized. . . . Apple Data Detectors [2] is another component architecture that supports automatic integration of tools. It works at the operating system level, using the selection mechanism that most Apple applications support. It allows the selection of a large area of text and recognizes all user-registered dtatypes in that selection. Users view suggested actions by pressing a modifier key and the mouse button. Like CyberDesk, it supports an arbitrary number of actions for each datatype."
if searching finds any second information related to the search term, performing the action using at least part of the second information, wherein the action is of a type depending at least in part on the type or types of the first information.	CyberDesk discloses, if searching finds any second information related to the search term, performing the action using at least part of the second information, wherein the action is of a type depending at least in part on the type or types of the first information. See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): "The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend's house but nobody is home. The system asks if he wants to check his friend's calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author's contact information in the contact manager, call the author, or send an e-mail to the author." See, e.g., CyberDesk Summary at 75 (including fig. 1): "The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a

Exhibit C-11

'843 Patent Claims	Disclosure
	phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy's phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood's contact information in the contact manager (d)." See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): "The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate 'ActOn' window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight 'Gregory Abowd' causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn't have Gregory's phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f)." See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): "The CyberDesk system has a simple but innovative architecture. It is based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in the form of suggestions to the user, via the ActOn Button Bar. It is through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below."

Exhibit C-11

’843 Patent Claims	Disclosure
	See, e.g., CyberDesk Technical Report at 3, cols. 1-2: “The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them.” See, e.g., CyberDesk Technical Report at 4, col. 2: “Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the ‘To:’ or ‘From:’ field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture.” See, e.g., CyberDesk Tehnical Note at 553, cols. 1-2: “Finally, the user’s selection, plus any extra type information generated by the conversion process, is observed by the ActOn integrating applet and a set of potential actions for that data is suggested. For example, a name is accepted as input by the Switchboard service, and so ActOn creates a button that suggests that integrating behaviour. Clicking on the button invokes the Switchboard service, completing the integration without

Exhibit C-11

’843 Patent Claims	Disclosure
	requiring any change to the functionality of either service and without an programming effort from the user.” See, e.g., CyberDesk Technical Report at 4, col. 2: “Data typing is used extensively in the interface declarations of the event sources and sinks that applications provide. The property field that corresponds to each interface declares the datatype/event that a component is interested in or can provide. The CyberDesk system takes advantage of the Java type system to do the data typing.” See, e.g., CyberDesk Technical Report at 4, col. 2: “Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the ‘To:’ or ‘From:’ field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture.” See, e.g., CyberDesk Technical Report at 5, col. 1: “So, when any component generates a String selection even, the StringToEmailAddress converter (and any other observers) are notified, and the converter attempts to conver the given String object to an EmailAddress object (while other converters attempt to convert the object to another CyberDesk type). In the above scenario, this conversion was done when the user selected the e-mail address. The system initially saw the selected data as a String but with this converter, it also saw it as an EmailAddress. This results in two related data selection events to arrive at the IntelliButton: one containing a string and one containing an EmailAddress. The IntelliButton will therefore seek integrating behaviour for both these types, allowing the user to access EmailAddress-relevant services where originally they wouldn’t have had the option.”

Exhibit C-11

’843 Patent Claims	Disclosure
	See, e.g., CyberDesk Technical Report at 5, col. 1: “Currently the list of CyberDesk types include: Date PhoneNumber, Mailing Address, Name, URL, and EmailAddress. If any of the conversions can be made, then the converter generates a second, but related, selection event containing the newly typed data and sends it to observing entities.” See, e.g., CyberDesk Technical Report at 5, cols. 1-2: “The CyberDesk framework was designed to be easily extensible. Simple extensions to CyberDesk include adding additional types, type converters, desktop services and network services. The real advantages with CyberDesk can be seen with more complex extensions that include adapting the behaviour of CyberDesk to individual use and creating more interesting integrating behaviour.” See, e.g., CyberDesk Technical Report at 5, col. 2: “An example converter is the StringToEmailAddress converter, which is a subclass of the ConversionApplet class. The code for this component and all components described in the paper can be viewed at http://www.cc.gatech.edu/fce/cyberdesk/samples. This converter looks at traditional ways of writing an e-mail address, and tries to map selected data to one of these ways. If it is successful, it returns an EmailAddress object. The ConversionApplet object is responsible for handling the ties to the CyberDesk framework.” See, e.g., CyberDesk Technical Report at 6, col. 1 (figure 3): “Screenshot of contact manager being used with CyberDesk. The user selects the string ‘Andy Wood’ in the –mail tool (a). CyberDesk offers some suggestions (b): search using AltaVista, look up a phone number using Switchboard (c), and look up the name in the contact manager (d).” See, e.g., CyberDesk Technical Report at 7, cols. 1-2: “The next task involved adding the services that recognized the appropriate types and created ‘ActOn’ buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above.” See, e.g., CyberDesk Technical Note at 553, col. 1 (including fig. 3): “When the user selects information displayed by one service, say some text from the e-mail message, the type converters try recursively to see if the data can be converted to other types used in the system (e.g., a name in Figure 3). In the case of plain text, this could be done by comparing the string to common formats for representing the various types; for

Exhibit C-11

'843 Patent Claims	**Disclosure**
	names you might use title firstname lastname, and similar patterns can be used for dates, URLs, e-mail and mailing addresses." See, e.g., CyberDesk Technical Report at 8, cols. 1-2: "CyberDesk contains some simple notions of context. It knows the application a user is working with and the data (both type and content) the user is interested in (via explicit selection with the mouse). But CyberDesk has shown the potential for supporting higher level context. For example, if an e-mail message contains information about a meeting, and the user selects the message content, a type converter could potentially convert the text to a Meeting object to be inserted in the user's Calendar Manager. Of course, retrieving context from arbitrary text is a very difficult problem being investigated by the AI learning community. But the power of CyberDesk supports the ability to use this higher level context, if available." See, e.g., CyberDesk Technical Report at 9, col. 2: "Related Work. Pandit and Kalbags Selection Recognition Agent [12] attempts to address the same issues as CyberDesk. Unlike CyberDesk, it uses a fixed datatype-action pair, allowing for only one possible action for each datatype recognized. . . . Apple Data Detectors [2] is another component architecture that supports automatic integration of tools. It works at the operating system level, using the selection mechanism that most Apple applications support. It allows the selection of a large area of text and recognizes all user-registered dtatypes in that selection. Users view suggested actions by pressing a modifier key and the mouse button. Like CyberDesk, it supports an arbitrary number of actions for each datatype." See, e.g., CyberDesk Technical Report at 3, cols. 1-2: "The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them." See, e.g., CyberDesk Tehnical Note at 553, cols. 1-2: "Finally, the

Exhibit C-11

'843 Patent Claims	**Disclosure**
	user's selection, plus any extra type information generated by the conversion process, is observed by the ActOn integrating applet and a set of potential actions for that data is suggested. For example, a name is accepted as input by the Switchboard service, and so ActOn creates a button that suggests that integrating behaviour. Clicking on the button invokes the Switchboard service, completing the integration without requiring any change to the functionality of either service and without an programming effort from the user." See, e.g., CyberDesk Technical Report at 4, col. 2: "Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the 'To:' or 'From:' field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture." See, e.g., CyberDesk Technical Report at 5, col. 1: "So, when any component generates a String selection even, the StringToEmailAddress converter (and any other observers) are notified, and the converter attempts to conver the given String object to an EmailAddress object (while other converters attempt to convert the object to another CyberDesk type). In the above scenario, this conversion was done when the user selected the e-mail address. The system initially saw the selected data as a String but with this converter, it also saw it as an EmailAddress. This results in two related data selection events to arrive at the IntelliButton: one containing a string and one containing an EmailAddress. The IntelliButton will therefore seek integrating behaviour for both these types, allowing the user to access EmailAddress-relevant services where originally they wouldn't have had the option."
Claim 8	

Exhibit C-11

'843 Patent Claims	Disclosure
A method according to claim 1, further comprising, providing a prompt for updating the information source to include the first information.	CyberDesk discloses providing a prompt for updating the information source to include the first information. See, e.g., CyberDesk Summary at 75 (including fig. 1): "The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy's phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood's contact information in the contact manager (d)." See, e.g., CyberDesk Technical Report at 3, cols. 1-2: "The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them." See, e.g., CyberDesk Technical Report at 8, col. 1: "Along the same line of thought, chaining can be used along with the concept of 'combining' to make services more powerful. . . . Combining in CyberDesk terms, is the ability to collect multiple data types for a piece of selected data, and bind them together, as needed, to create multiple meta-objects. These meta-objects could be used to perform substantially more powerful actions. Taking the above example of a user reading an appointment in her calendar, assume that there are multiple services capable of chaining, so a URL selection event is created, a Date selection event is created, a MailingAddress selection event is created, etc. There is potentially enough information to create a new entry in the Contact Manager. The ability to assimilate this widespread information in a compact entity is very powerful for a user. It's this same ability that would allow us to combine time and a name to search a friend's schedule as seen in the user scenario."

'843 Patent Claims	Disclosure
	See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): "The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend's house but nobody is home. The system asks if he wants to check his friend's calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author's contact information in the contact manager, call the author, or send an e-mail to the author." See, e.g., CyberDesk Technical Report at 4, cols. 1-2: "All of the desktop applets currently being used in CyberDesk (2 e-mail browsers, contact manager, 2 calendar managers/schedulers, scratchpad) were previously written by other Georgia Tech students." See, e.g., CyberDesk Technical Report at 5, col. 2—6, col. 1 (including fig. 3): "The example below is the wrapper for the Contact Manager (see Figure 3), and it extends the ContactApplet class (the original application class). . . . Lookup an entry for the name in the ContactManager." See, e.g., CyberDesk Technical Report at 6, col. 1 (figure 3): "Screenshot of contact manager being used with CyberDesk. The user selects the string 'Andy Wood' in the –mail tool (a). CyberDesk offers some suggestions (b): search using AltaVista, look up a phone number using Switchboard (c), and look up the name in the contact manager (d)." See, e.g., CyberDesk Technical Report at 7, cols. 1-2: "The next task involved adding the services that recognized the appropriate types and created 'ActOn' buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above."

Exhibit C-11

'843 Patent Claims	Disclosure
	See, e.g., CyberDesk Technical Report at 7, col. 2—8, col. 1: "The extensions described so far are fairly simplistic. They deal with adding more suggestions for the user. More interesting are extensions which add more powerful suggestions for the user. Chaining is an example of this type of extension. It extends CyberDesk's type converting ability by using network services as type converters, to allow for increased integrating behaviour. For example, a user is reading an appointment in her calendar manager, and selects the name of the person she's supposed to be meeting. As an experienced user, she expects to be presented with a list of all possible services that can use a Name: search for a phone number, mailing address, look up in the contact manager, search name on the WWW, etc. However, by using chaining, more powerful suggestions can be had. The WhoWhereEmail network described earlier, takes a name as input and returns a WWW browser showing a list of possible e-mail addresses corresponding to that name. If we make the assumption (not always a good one) that the first e-mail address returned in the list is the correct one, we can now use this service to convert the name to an e-mail address. The service now creates a related EmailAddress selection event, and the user is supplied with all possible suggestions for both a Name and an EmailAddress." See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): "The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate 'ActOn' window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight 'Gregory Abowd' causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn't have Gregory's phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f)." If the Judge or Jury finds that CyberDesk does not anticipate this limitation, then CyberDesk still renders the claim obvious for the reasons discussed in Exhibit F, Table 9.
Claim 13	
A method according to claim 1,	CyberDesk discloses the method of claim 1 wherein the user command is

Exhibit C-11

’843 Patent Claims	Disclosure
wherein the user command is the only command from a user necessary to initiate performing the operation.	the only command from a user necessary to initiate performing the operation. See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): “The user highlights some text in the window of one service, and CyberDesk determines the tpe of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate ‘ActOn’ window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight ‘Gregory Abowd’ causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn’t have Gregory’s phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f).” See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): “The CyberDesk system has a simple but innovative architecture. It is based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in the form of suggestions to the user, via the ActOn Button Bar. It is through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below.” See, e.g., CyberDesk Technical Report at 3, cols. 1-2: “The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a

'843 Patent Claims	Disclosure
	component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them." See, e.g., CyberDesk Tehnical Note at 553, cols. 1-2: "Finally, the user's selection, plus any extra type information generated by the conversion process, is observed by the ActOn integrating applet and a set of potential actions for that data is suggested. For example, a name is accepted as input by the Switchboard service, and so ActOn creates a button that suggests that integrating behaviour. Clicking on the button invokes the Switchboard service, completing the integration without requiring any change to the functionality of either service and without an programming effort from the user." See, e.g., CyberDesk Technical Report at 3, col. 2: "The ActOn Button Bar, as described before, is simply the user interface for the integrating IntelliButton. We chose to keep the interface separate from the actual integrating functionality to all easier experimentation with alternative interfaces. Currently, the interface is very simplistic. It is a dynamically generated list of buttons, where each button corresponds to a particular service that can be executed based on an user-generated event and its corresponding data. The list of buttons is provided by the IntelliButton. . . . For example: Send e-mail to this EmailAddress using Netscape; Search for a string on the Web using AltaVista." See, e.g., CyberDesk Technical Report at 4, col. 1: "With this interface, this search would be suggested by the IntelliButton whenever a text string is the target of a selection (assuming the component in which the text selection is done, supports the 'select' interface)." See, e.g., CyberDesk Technical Report at 4, col. 2: "Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the 'To:' or 'From:' field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar,

'843 Patent Claims	Disclosure
	which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture." See, e.g., CyberDesk Technical Report at 5, col. 1: "So, when any component generates a String selection even, the StringToEmailAddress converter (and any other observers) are notified, and the converter attempts to conver the given String object to an EmailAddress object (while other converters attempt to convert the object to another CyberDesk type). In the above scenario, this conversion was done when the user selected the e-mail address. The system initially saw the selected data as a String but with this converter, it also saw it as an EmailAddress. This results in two related data selection events to arrive at the IntelliButton: one containing a string and one containing an EmailAddress. The IntelliButton will therefore seek integrating behaviour for both these types, allowing the user to access EmailAddress-relevant services where originally they wouldn't have had the option." See, e.g., CyberDesk Technical Report at 6, col. 2: "The following is an example network service: WhoWhereEmail. It is a gateway to the WhoWhere network service available on the web. When a name is input into the web service, a list of possible e-mail addresses corresponding to that name is returned. The CyberDesk network service takes a Name object, inputs it into the service and displays the results in a web browser." See, e.g., CyberDesk Technical Report at 7, cols. 1-2: "The next task involved adding the services that recognized the appropriate types and created 'ActOn' buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above."

Exhibit C-11

’843 Patent Claims	**Disclosure**
	See, e.g., CyberDesk Technical Report at 9, col. 1: “Perhaps the biggest limitation of the system is the user interface implemented by the ActOn Button Bar. It consists of a window that displays a long list of suggested user actions. It is clear that the number of possible suggestions could quickly become overwhelming to the user. We are currently looking at different ways to adapt the interface to initially show actions that the user is likely to take, but provide a way for the user to see other possible actions as well. We are also looking at different presentation methods for the suggestions, including pop-up hierarchical menus and document lenses [4].” See, e.g., CyberDesk Technical Report at 9, col. 2: “Related Work. Pandit and Kalbags Selection Recognition Agent [12] attempts to address the same issues as CyberDesk. Unlike CyberDesk, it uses a fixed datatype-action pair, allowing for only one possible action for each datatype recognized. . . . Apple Data Detectors [2] is another component architecture that supports automatic integration of tools. It works at the operating system level, using the selection mechanism that most Apple applications support. It allows the selection of a large area of text and recognizes all user-registered dtatypes in that selection. Users view suggested actions by pressing a modifier key and the mouse button. Like CyberDesk, it supports an arbitrary number of actions for each datatype.”
Claim 15	
A method according to claim 1, further comprising, if searching results in a plurality of distinct instances of second information, displaying such instances to enable user selection of one of them for use in performing the action.	CyberDesk discloses the method of claim 1 further comprising, if searching results in a plurality of distinct instances of second information, displaying such instances to enable user selection of one of them for use in performing the action. See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): “The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend’s house but nobody is home. The system asks if he wants to check his friend’s calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the

Exhibit C-11

’843 Patent Claims	Disclosure
	author’s contact information in the contact manager, call the author, or send an e-mail to the author.” See, e.g., CyberDesk Summary at 75 (including fig. 1): “The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy’s phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood’s contact information in the contact manager (d).” See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): “The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate ‘ActOn’ window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight ‘Gregory Abowd’ causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn’t have Gregory’s phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f).” See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): “The CyberDesk system has a simple but innovative architecture. It is based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in

Exhibit C-11

’843 Patent Claims	Disclosure
	the form of suggestions to the user, via the ActOn Button Bar. It is through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below.” See, e.g., CyberDesk Technical Report at 3, cols. 1-2: “The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them.” See, e.g., CyberDesk Technical Report at 4, col. 2: “Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the ‘To:’ or ‘From:’ field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture.”

Exhibit C-11

'843 Patent Claims	**Disclosure**
	If the Judge or Jury finds that CyberDesk does not anticipate this limitation, then CyberDesk still renders the claim obvious for the reasons discussed in Exhibit F, Table 18.
Claim 17	
A method according to claim 1, wherein the information source is associated with the second computer program and is available through the computer.	CyberDesk discloses the method of claim 1 wherein the information source is associated with the second computer program and is available through the computer. See, e.g., CyberDesk Technical Report at 1, col. 2: "The components in CyberDesk treat all data uniformly, regardless of whether the data came from a locally running application or from a service running on the World Wide Web (WWW). The services and applications themselves can be running anywhere, meeting CyberDesk's goal of providing ubiquitous access to services." See, e.g., CyberDesk Technical Report at 7, cols. 1-2: "The next task involved adding the services that recognized the appropriate types and created 'ActOn' buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above."
Claim 18	
A method according to claim 1, wherein performing the action includes causing insertion of at least part of the second information into the document.	CyberDesk discloses the method of claim 1 wherein performing the action includes causing insertion of at least part of the second information into the document. See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): "The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend's house but nobody is home. The system asks if he wants to check his friend's calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author's contact information in the contact manager, call the author, or

Exhibit C-11

’843 Patent Claims	Disclosure
	send an e-mail to the author.” See, e.g., CyberDesk Summary at 75 (including fig. 1): “The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy’s phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood’s contact information in the contact manager (d).” See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): “The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate ‘ActOn’ window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight ‘Gregory Abowd’ causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn’t have Gregory’s phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f).” See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): “The CyberDesk system has a simple but innovative architecture. It is based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in the form of suggestions to the user, via the ActOn Button Bar. It is

Exhibit C-11

'843 Patent Claims	**Disclosure**
	through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below." See, e.g., CyberDesk Technical Report at 3, cols. 1-2: "The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them." See, e.g., CyberDesk Technical Report at 4, col. 2: "Initially, we hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the 'To:' or 'From:' field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture."
Claim 19	

Exhibit C-11

’843 Patent Claims	Disclosure
A method according to claim 1, wherein performing the action includes causing insertion of at least part of the second information into the document by the first computer program.	CyberDesk discloses the method of claim 1 wherein performing the action includes causing insertion of at least part of the second information into the document. See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): “The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend’s house but nobody is home. The system asks if he wants to check his friend’s calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author’s contact information in the contact manager, call the author, or send an e-mail to the author.” See, e.g., CyberDesk Summary at 75 (including fig. 1): “The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy’s phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood’s contact information in the contact manager (d).” See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): “The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate ‘ActOn’ window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight ‘Gregory Abowd’ causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b).

Exhibit C-11

'843 Patent Claims	**Disclosure**
	Anind discovers that he doesn't have Gregory's phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f)." See, e.g., CyberDesk Technical Report at 2, cols. 1-2 (including figures): "The CyberDesk system has a simple but innovative architecture. It is based on an event-driven model, where components act as even sources and/or event sinks. Events, in this current version, are generated from explicit user interaction with the system. The system consists of five core components: the Locator, the IntelliButton, the ActOn Button Bar, the desktop and network services, and the type converters. The Locator maintains the registry of event sources and sinks. This allows the IntelliButton to automatically find matches between event sources and even sinks based on a given input event, a task normally required of the system or service designer. The IntelliButton displays the matches in the form of suggestions to the user, via the ActOn Button Bar. It is through the ActOn Button Bar that the user accesses the integrating functionality of CyberDesk. The services are the even sources and sinks themselves, and are the tools the user ultimately wants to use. The type converters provide more powerful integrating behavior by converting given events into other events, allowing for a greater number of matches. The five components are discussed in greater detail below." See, e.g., CyberDesk Technical Report at 3, cols. 1-2: "The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them." See, e.g., CyberDesk Technical Report at 4, col. 2: "Initially, we

Exhibit C-11

'843 Patent Claims	**Disclosure**
	hardcoded applications to generate events for different data types. For example, the e-mail browser declares that it can generate String selection events when text is highlighted, but also EmailAddress selection events when the 'To:' or 'From:' field in an e-mail message is selected. When EmailAddress selection events were generated, they were passed through the CYberDesk system, as described before, to the ActOn Button Bar, which displayed services that could consume EmailAddress selection events (e.g., Send an E-mail to this E-mail Address using Netscape). . . . Consequently, we chose to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment our e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an EmailAddress object and create an EmailAddress selection event rather than just a String selection event. But, rather than just giving this type conversion capability to the e-mail browser, we wanted to add that ability to the system once, and all it to be used in every application where e-mail addresses might appear. We took the type detection ability out of the individual applications and created type converters, an independent and extensible layer in the architecture."
Claim 23	
At least one non-transitory computer readable medium encoded with instructions which, when loaded on a computer, establish processes for finding data related to the contents of a document using a first computer program running on a computer, the processes comprising:	CyberDesk discloses at least one non-transitory computer readable medium encoded with instructions which, when loaded on a computer, establish processes for finding data related to the contents of a document using a first computer program running on a computer. See, e.g., CyberDesk Technical Report at 1, col. 1: "Current software suites suffer from problems due to poor integration of their individual tools. They require the designer to think of all possible integrating behaviours and leave little flexibility to the user. In this paper, we discuss CyberDesk, a component software framework that automatically integrates desktop and network services, requiring no integrating decisions to be made by the tool designers and giving total control to the user." See, e.g., CyberDesk Technical Report at 1, col. 1: "In response, software companies have been adopting the notion of component software: using small software modules as building blocks for a larger application. While there are many competing standards (OLE [11], Active X [10], Java Beans [6], OpenDoc [1]), the prevailing view is to provide a framework which programmers and sophisticated users can build upon to create desired application suites." See, e.g., CyberDesk Technical Report at 1, col. 2: "In this paper, we present CyberDesk system, a component software framework that relieves most of the burden of integrating services from both the designer

’843 Patent Claims	Disclosure
	of individual services and the end user, provides greater flexibility to the user, and automatically suggests how independent services can be integrated in interesting ways.” See, e.g., CyberDesk Technical Report at 1, col. 2: “CyberDesk is a component-based framework written in Java, that supports automatic integration of desktop and network services [16]. The framework is flexible, and can be easily customized and extended.” See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): “The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend’s house but nobody is home. The system asks if he wants to check his friend’s calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author’s contact information in the contact manager, call the author, or send an e-mail to the author.” See, e.g., CyberDesk Summary at 75 (including fig. 1): “The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy’s phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood’s contact information in the contact manager (d).” See, e.g., CyberDesk Technical Report at 4, cols. 1-2: “All of the desktop applets currently being used in CyberDesk (2 e-mail browsers, contact manager, 2 calendar managers/schedulers, scratchpad) were previously written by other Georgia Tech students.” See, e.g., CyberDesk Technical Report at 5, col. 2—6, col. 1 (including

Exhibit C-11

'843 Patent Claims	Disclosure
	fig. 3): "The example below is the wrapper for the Contact Manager (see Figure 3), and it extends the ContactApplet class (the original application class). . . . Lookup an entry for the name in the ContactManager." See, e.g., CyberDesk Technical Report at 6, col. 1 (figure 3): "Screenshot of contact manager being used with CyberDesk. The user selects the string 'Andy Wood' in the –mail tool (a). CyberDesk offers some suggestions (b): search using AltaVista, look up a phone number using Switchboard (c), and look up the name in the contact manager (d)." See, e.g., CyberDesk Summary at 75 (including fig. 1): "The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy's phone number and mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood's contact information in the contact manager (d)." *See also* preamble in Claim 1.
displaying the document electronically using the first computer program;	CyberDesk discloses displaying the document electronically using the first computer program. *See* corresponding limitation in Claim 1.
while the document is being displayed, analyzing, in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information;	CyberDesk discloses, while the document is being displayed, analyzing, in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information. *See* corresponding limitation in Claim 1.
retrieving the first information;	CyberDesk discloses retrieving the first information. *See* corresponding limitation in Claim 1.
providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, the operation comprising (i) performing a search using at	CyberDesk discloses providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, the operation comprising (i) performing a search using at least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the search term in an information source external to the document, wherein

Exhibit C-11

'843 Patent Claims	**Disclosure**
least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the search term in an information source external to the document, wherein the specific type or types of second information is dependent at least in part on the type or types of the first information, and (ii) performing an action using at least part of the second information;	the specific type or types of second information is dependent at least in part on the type or types of the first information, and (ii) performing an action using at least part of the second information. *See* corresponding limitation in Claim 1.
in consequence of receipt by the first computer program of the user command from the input device, causing a search for the search term in the information source, using a second computer program, in order to find second information related to the search term; and	CyberDesk discloses, in consequence of receipt by the first computer program of the user command from the input device, causing a search for the search term in the information source, using a second computer program, in order to find second information related to the search term. *See* corresponding limitation in Claim 1.
if searching finds any second information related to the search term, performing the action using at least part of the second information, wherein the action is of a type depending at least in part on the type or types of the first information.	CyberDesk discloses, if searching finds any second information related to the search term, performing the action using at least part of the second information, wherein the action is of a type depending at least in part on the type or types of the first information. *See* corresponding limitation in Claim 1.
Claim 30	
At least one non-transitory computer readable medium according to claim 23, the instructions establishing processes comprising:	CyberDesk discloses the non-transitory computer readable medium according to claim 23.
providing a prompt for updating the information source to include the first information.	CyberDesk discloses providing a prompt for updating the information source to include the first information. See, e.g., CyberDesk Summary at 75 (including fig. 1): "The user receives an e-mail message (see Figure 1) with the name Andy Wood in it. She highlights the name with her mouse (a) and is shown a list of suggested actions she can perform (b). This list includes searching for the selected text using the AltaVista web search service, looking up a phone number for the selected name using the Switchboard web service, or looking up the selected name in the desktop contact manager. The user chooses the second option and retrieves Andy's phone number and

Exhibit C-11

'843 Patent Claims	Disclosure
	mailing address from the web (c). She wants to update her contact information for Andy, so she chooses the last option which loads Andy Wood's contact information in the contact manager (d)." See, e.g., CyberDesk Technical Report at 3, cols. 1-2: "The IntelliButton component is really the core of the CyberDesk system, as it provides the automatic integrating behaviour. . . . So when a component generates an event, it notifies the IntelliButton and any other components that have expressed interest. . . . It uses simple type checking to identify potential services that the user may wish to call upon to operate on the data associated with the event. The matches are displayed to the user via the ActOn Button Bar, from which the user can select any or none of the integrating services suggested. If the user does choose one of the integrating services, the IntelliButton is notified and it access the correct service passing the associated data and event as parameters. In the above scenario, when the user highlighted the e-mail address, the IntelliButton used that even information to determine what services were available (send an e-mail, save the contact information, etc.) and suggested them." See, e.g., CyberDesk Technical Report at 8, col. 1: "Along the same line of thought, chaining can be used along with the concept of 'combining' to make services more powerful. . . . Combining in CyberDesk terms, is the ability to collect multiple data types for a piece of selected data, and bind them together, as needed, to create multiple meta-objects. These meta-objects could be used to perform substantially more powerful actions. Taking the above example of a user reading an appointment in her calendar, assume that there are multiple services capable of chaining, so a URL selection event is created, a Date selection event is created, a MailingAddress selection event is created, etc. There is potentially enough information to create a new entry in the Contact Manager. The ability to assimilate this widespread information in a compact entity is very powerful for a user. It's this same ability that would allow us to combine time and a name to search a friend's schedule as seen in the user scenario." See, e.g., CyberDesk Technical Report at 1, col. 2—2, col. 1 (including figures): "The user walks to a grocery store, and the system asks if he wants to see his shopping list, get more information about the grocery store, or get directions to his house. The user chooses the grocery list and goes shopping. He walks to a friend's house but nobody is home. The system asks if he wants to check his friend's calendar, contact him via e-mail or phone, or get directions to go home. The user chooses the first option and the system tells him that his friend is at work. So, he chooses the second option, sends his friend an email saying that he stopped by, and starts walking home. On the way home, the system

Exhibit C-11

'843 Patent Claims	Disclosure
	notifies him that he has received an e-mail from his friend. The user reads the e-mail (see Figure 1 below) which has information on a new book written by his favourite author. The e-mail contains a Web site address and an e-mail address for the author. The user highlights the e-mail address (a) and the system gives him some suggestions (b) on what he can do: search for more information on the author, put the author's contact information in the contact manager, call the author, or send an e-mail to the author." See, e.g., CyberDesk Technical Report at 4, cols. 1-2: "All of the desktop applets currently being used in CyberDesk (2 e-mail browsers, contact manager, 2 calendar managers/schedulers, scratchpad) were previously written by other Georgia Tech students." See, e.g., CyberDesk Technical Report at 5, col. 2—6, col. 1 (including fig. 3): "The example below is the wrapper for the Contact Manager (see Figure 3), and it extends the ContactApplet class (the original application class). . . . Lookup an entry for the name in the ContactManager." See, e.g., CyberDesk Technical Report at 6, col. 1 (figure 3): "Screenshot of contact manager being used with CyberDesk. The user selects the string 'Andy Wood' in the –mail tool (a). CyberDesk offers some suggestions (b): search using AltaVista, look up a phone number using Switchboard (c), and look up the name in the contact manager (d)." See, e.g., CyberDesk Technical Report at 7, cols. 1-2: "The next task involved adding the services that recognized the appropriate types and created 'ActOn' buttons for them. We added two services (NewtonNames and NewtonNotes) which, respectively, request contact information from the Newton about the selected name and request notes from the Newton containing the selected text in the body or title. Adding these services was quite simple (see Figure 4), requiring only the implementation of the ServiceApplet methods described above." See, e.g., CyberDesk Technical Report at 7, col. 2—8, col. 1: "The extensions described so far are fairly simplistic. They deal with adding more suggestions for the user. More interesting are extensions which add more powerful suggestions for the user. Chaining is an example of this type of extension. It extends CyberDesk's type converting ability by using network services as type converters, to allow for increased integrating behaviour. For example, a user is reading an appointment in her calendar manager, and selects the name of the person she's supposed to be meeting. As an experienced user, she expects to be presented with a list of all possible services that can use a Name: search for a phone number, mailing address, look up in the contact manager, search name on

’843 Patent Claims	Disclosure
	the WWW, etc. However, by using chaining, more powerful suggestions can be had. The WhoWhereEmail network described earlier, takes a name as input and returns a WWW browser showing a list of possible e-mail addresses corresponding to that name. If we make the assumption (not always a good one) that the first e-mail address returned in the list is the correct one, we can now use this service to convert the name to an e-mail address. The service now creates a related EmailAddress selection event, and the user is supplied with all possible suggestions for both a Name and an EmailAddress.” See, e.g., CyberDesk Technical Note at 552, col. 2—553, col. 1 (including figs. 1, 2): “The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate ‘ActOn’ window. For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlight ‘Gregory Abowd’ causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn’t have Gregory’s phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c). Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that reference URL (f).” If the Judge or Jury finds that CyberDesk does not anticipate this limitation, then CyberDesk still renders the claim obvious for the reasons discussed in Exhibit F, Table 9.