Bibliography

[1] Edie Bender. *Presentation to Board of Trustees.* Penn State, July 1991. Photocopies of slides used in presentation.

[2] Joanne Costello. Help desk database. Technical report, Massachussetts Institute of Technology, April 1991.

[3] Heikki Hämmäinen, Eero Eloranta, and Jari Alasuvanto. Distributed form management. *ACM Transations on Information Systems*, January 1990.

[4] R.M. Huberman, M. Tessler, and W.H. Cameron. A generic forms handling tool and its application to network trouble management. In *IEEE International Conference on Communications*, 1989.

[5] James A. Larson and Jennifer Wallick. Tools for forms administrators. In *IEEE Workshop on Languages for Automation*, 1984.

[6] Plato. *Selections*, page 214. Charles Scribner's Sons, 1927.

[7] Requirements definition: Electronic approval/routing system. Technical Report CSD0516, University of California, 1991.

Appendix 1

Terms used in this paper

Term	Meaning
administrator	The person responsible for the day-to-day maintance of the Phaedo server and the database as a whole.
author	The person who initially designs a form template. The author has access to special tools for this purpose, which are not available to the average user.
form	Unless otherwise modified, this means "form instance."
form description	Describes how the form should look when it is displayed on the screen. This contains a listing of the different input fields on the form, the fields' default values, read-only text to aid the user in filling out the form, and additional internal information on how to display the fields.
form instance	A form template which has been filled out and is moving through the system. There is only one template for each type of form, but there can be many different instances.
form template	The abstract specification of a particular type of form, composed of the form description and the route description.
path	The set of stops travelled by a single form instance
route	The collection of all stops legal for a specific type of form, along with the allowed connections between the stops.
route description	A list of the different stops along the lifecycle of this type of form, and a list of who is responsible for the form by default for each step.
stop	A state in which a form can be, such as "pending supervisor approval."
user	Someone who uses the form system just to fill out blank forms or use and forward existing forms.

This work has already been begun. We plan to use their expectations and requirements, research into similar systems, and our own beliefs as to what will make the system most generally usable, to chart the further design of Phaedo.

- **Must research and acquire a DBMS.** The current prototype of Phaedo uses a toy Database Management System based on the UNIX file system. Before the system can move into common use, a "real" DBMS must be decided on and obtained for use. It is likely that this will be Ingress, but we must ensure that a smooth migration path to a different database system remains open.

- **Phaedo must be culturally accepted.** Phaedo will fail unless it becomes a commonly accepted component of day-to-day work at the Institute. At a minimum, the following criteria will have to be true before this can happen.

  - **Must suit community requirements.** As the development of Phaedo continues, the designers and implementers must maintain strong ties to probable users of the system, and ensure that the system, as planned and implemented, will meet the needs of the users.
  - **Must be championed from beyond DCNS.** A system such as Phaedo cannot be "pushed." We can make it publicly available, but cannot force external organizations to use it in their work. We must cultivate "champions" of the system from outside Distributed Computing and Network Services, who will "pull" it into the work done by their organizations.
  - **Excellent user support must be provided.** Our job will not be finished once the code is completed and the executables installed. We must plan on providing excellent support to those who use the system. At least two levels of support must be provided: First, the users of Phaedo must be helped in using forms already in the system; and second, the authors of these forms must be supported in maintaining their forms and in designing new electronic forms and placing them into the system. Support of end-users will be the job of the authors, and support of the authors will fall to the builders and maintainers of Phaedo itself. At all times, we must monitor the use of the system and make certain that it actually supports the needs of the users. If it falls short, we must stand ready to revise it so that it meets the measure.

Conclusion

Constructing an integrated administrative system such as Penn State's IBIS and EASY systems[1] is not currently possible at MIT, due to the fragmentation of administrative information and databases. While we cannot immediately eliminate all administrative paperwork, we *can* provide a form routing system with sufficient generality that it can serve as a path towards this ultimate goal, while being a tool of significant and immediate use in other areas of the Institute. We expect that Phaedo will be this tool.

The next step in this project is to identify some potential users of the package and find out what they expect from an electronic forms system.

and immediately notify the user if it appears something is amiss. Phaedo may be able to dynamically configure a form, based on data it receives from querying another system.

- **Sender of data.** If an external system cooperates with Phaedo's protocol, this system may use Phaedo as an input source. Phaedo will be able to deliver data in a format designed to be readily used by whatever system it is interacting with.

"Desirements"

The items listed below, while not critically important, will greatly enhance the utility of Phaedo. We intend to eventually include as many of them as possible.

- **Conditional Routing.** Once a user has filled out a form, the system may be able to select automatically among various destinations for it, depending on the information on the form.
- **Real Time Notification.** We would like Phaedo to have the ability to notify designated persons of "interesting" events, including values on the form being changed, the form moving from stop to stop, and possibly a timer expiring if the form has not been touched within a certain time. This notification may take the form of sending email or zephyrgrams, or using some other appropriate means of communication. The amount of notification should be selectable by the user on a per-instance basis, and masked against the amount of notification made available by the form's author.
- **Waiting/Active Form States.** It would be convenient if the system were able to distinguish between a form waiting for someone to act on it (effectively, in a queue or a mailbox) and a form which has been taken and is being acted on.
- **Cloning.** It would be useful if a user were able to make a "clone" of a form already in the system, by using its current contents as the initial values for fields in a new form. It may not be necessary for these forms to be of the same type.
- **Complex approval lists.** In some cases, saying that a single person has authority for approving a form at a given stop may not be sufficient. In these cases, supporting operations such as "and," "or," and "X out of Y" on approval lists would be useful.

Issues to be resolved

The following items are issues which must be considered and resolved to ensure that Phaedo fits into the environment at MIT:

- **Configurable Cancellation/Revocation Rules.** For some types of forms, users may be able to cancel forms they have originated, or retract forms they have approved. The route description for each type of form will include rules describing which activities are allowed for this form.

- **Linking of Forms.** It should be possible to link together related forms. This may be as simple as providing a type of field which gives a pointer to a related form, or as complex as special routing rules which only allow a form to advance if its partner has reached a given state.

- **Hardcopy.** It must be possible to produce a paper copy of a form at any time.

- **Proxies.** It must be possible for a user to delegate a proxy to handle his or her forms. It would be desirable for the mechanism supporting proxies be intelligent enough to forward incoming forms to different agents depending on information contained in the forms, though this is not a requirement.

- **Real Time Query Support.** The users must be able to check on the status of individual forms which are traveling within the system, subject to access control. Also, users must be able to search through the form database (again, subject to access control) and retrieve an "interesting" set of forms, based on criteria which they supply.

- **Statisticial Logging.** The system should maintain logs of its usage and make these data available for statistical analysis. As with the audit trail, it must be possible to specify for each type of form the amount of data gathered (if any) and how access to this data is controlled.

- **Group Form Ownership.** The system must be able to support groups of users and allow either an individual or a group to own a form.

- **Incremental Growth.** It must be possible to replace a small portion of an existing process with Phaedo, and gradually increase Phaedo's portion of the process with time.

- **Interaction With Other Systems.** Phaedo must be able to communicate with a variety of systems external to itself. This will require writing communication protocols, designing interface modules for both Phaedo and the individual external databases, and bringing the databases onto the network, if they are not already.

  - **Receiver of data.** Phaedo will be able to issue queries for data such as the name of a person's supervisor, a person's phone number and office, and so forth, and provide these as default values on blank forms. For example, a form requiring approval of a person's supervisor might be able to automatically fill in the supervisor's name as the next person to whom the form should go. It will also be able to run sanity checks on user-input data,

intoshes, and DOS machines. In addition, a non-graphical interface must be available for dialup purposes, and for use from other systems which do not support a windowing interface. All forms must be available from all types of supported workstations. It must be possible to incrementally support new platforms without substantial revision of the system.

- **Authentication.** The system must have the ability to prove that users of the system are who they purport to be. It should be possible to "plug in" authentication modules providing different levels of security, as deemed appropriate for individual forms. Although Phaedo need not be impregnable, it should always be at least as secure as the manual systems currently in place.

- **Access Control.** The system must be able to control access to sensitive data, both on the forms and internal to the system itself. Access control down to the level of individual fields on a form must be supported, and it must be possible for access to these fields to change as the form moves through different stops on its route.[6]

- **Audit Trail.** The system must support the maintaining of a history of changes which have been made to forms in its custody. How much is logged, and how access to this information is controlled, will be specified by individual forms.

- **Formatted Data Entry.** The system must provide a variety of different types of input fields, including at a minimum free-form text, numeric fields, multiple-choice fields, and boolean fields.

- **Generic Form Description Language.** It must be possible for new types of forms to be added, and existing form templates[7] revised, without the need for modifying Phaedo itself. This will probably require that the system maintain a database of form descriptions[8] in a human-readable form description language.

- **Built-In Help.** The system's form description language must provide a means for adding help text, both to the form as a whole and to individual input fields.

- **Configurable Form Routing.** Each type of form will have a route associated with it. For example, the route for one type of form could begin at the stop "Being Requested," have "Pending Supervisor Approval," as the next stop, and then branch to either "Approved" or "Rejected." The system must provide a human-readable language for describing these routes, and it must be possible to change the routes without revising Phaedo itself.

[6] A *route* is the collection of all stops legal for a specific type of form, along with the allowed connections between the stops. For those with a Computer Science theory background, the route resembles a finite state automaton, with each stop being a state, and the list of paths between stops representing the transitions.

[7] A *form template* is the abstract specification of a particular type of form, composed of the form description and the route description.

[8] A *form description* describes how a form should look when it is displayed on the screen. It contains a listing of the different input fields on the form, the fields' default values, read-only text to aid the user in filling out the form, and additional internal information on how to display the fields.

connection request uses at least half-a-dozen) this can provide a substantial time savings.

The typical path[5] of a form will be part of each form's definition, making it possible for the forms to "know" the identity of the person to whom they should next be assigned and route themselves automatically to this destination.

Provides easier access to forms

Blank electronic forms will never be out of stock or obsolete. When electronic forms are revised or updated, the most current version will be immediately available to all users, and there will be no collection of obsolete blank forms to be disposed of. Furthermore, eliminating the paper itself saves on the cost of printing and storing physical forms.

Electronic forms will be available at any hour, on any day of the week. Once a form has been filled out and entered into the system, the person who filled it out, or any other authorized agent, will be able to query its status electronically without regard for whether or not the office servicing the request is open.

To provide an audit trail in the absence of physical forms, Phaedo will be able to place its inactive forms into a searchable archive.

Can talk to other information systems

We plan to make Phaedo able to communicate with other databases, both as a sender and a receiver of data.

Because Phaedo will collect information from its users in a structured manner, it will be able to deliver data to other systems in a well-defined format and ready to use.

As a receiver, it will be able to query existing databases for some field values, so that these fields can be automatically filled in and redundant data entry eliminated. By reducing the need for reentering data which is already in a database, such as a person's address and department number, many typographical errors can be avoided and integrity of data between databases can be maintained.

Requirements

We believe that the following features are necessary in a usable form routing system.

- **Extendable Multiplatform Accessibility.** The system must be usable from a variety of platforms, such as Athena workstations, Mac-

[5] A *path* is the set of stops travelled by a single form instance.

are required for an electronic form routing system. Appendicies include a glossary of terms used in this document, a bibliography of research papers and other documents which were studied during the construction of the Phaedo prototype, and a diagram suggested by the DMR "Productivity Plus" development methodology.

Benefits of electronic forms

We outline below some of the advantages electronic forms have over paper ones, using the MITnet connection request as an example. In the existing system, when a request comes in, it is written down on a paper form which is then handed off between various persons and groups. As the form travels, attachments are added (such as a site survey) and other processes are triggered (such as email requesting an IP address.)

Easier to use

It is faster and, we expect, easier to fill out forms electronically.

For example, on a windowing workstation, it will be possible for the person filling out a form to use standard cut-and-paste operations to take values displayed in one window and paste them into a blank form in another window. It may even possible for Phaedo to automatically fill in some portions of a form, provided that other databases can be queried during the filling-out process. For example, given a client's name, the helpdesk system would be able to automatically provide the matching department, phone number, and campus address.

The forms themselves will be interactive, and will aid the user in filling them out. Help text can be attached to the form as a whole, and also to individual fields within the form, and can be called up by the user as the form is being filled out.

The input fields will be somewhat intelligent. To a limited extent, they will recognize what sort of data they should contain, and will reject illegal input. For example, a field such as "Number of machines to be networked" could disallow alphabetic characters.

Perhaps most important from a user interface standpoint, Phaedo will provide a common user interface for all the forms it maintains, so that a user who has learned how to manipulate one form in the system can now manipulate all of them.

Transmits information more quickly

Transmitting the forms electronically allows for the elimination of manual handling of physical pieces of paper, thus making it possible for a form to move more quickly. For forms which require several stops (the MITnet

Introduction

Phaedo[1] is a project, currently in the requirements stage, being developed by the DCNS-Apps group. It is intended to provide a system with which users can manipulate electronic forms[2] much as they currently do paper forms. It will be possible for a typical user[3] to request a blank form from various possibilities, fill in this form on his or her computer screen, and forward it electronically to the next stop.[4] The person to whom the form is assigned can check it out of the system, read and manipulate the values on it, and either send it electronically to another user or permanently remove it from the system.

This project grew out of the development of a system to track help desk service requests,[2] as it became obvious that a generic system for routing electronic forms could be used in many different sites within the Institute.

Electronic forms have several advantages over their paper counterparts. They're never out of stock, and they're much easier to recycle! More seriously, they offer significant advantages in terms of data integrity, speed of movement between people, and the provision of a common interface to a variety of different forms.

Some electronic form systems, notably for requisitions and timecards, already are in successful use within the Institute. However, these are extremely specialized systems, each for a single use, and they do not communicate with other databases and information systems beyond their own domain.

With Phaedo, we plan to step beyond the limitations of the existing systems. Phaedo will offer a generic mechanism for creating an interactive electronic version of almost any existing form. We expect it to serve as the "glue" which will bind together a variety of existing, currently isolated, information systems, while providing a consistent user interface for access to these systems.

The remainder of this paper will describe a number of advantages of using electronic forms over paper ones, and then list the features we believe

[1] From Plato's *Phaedo* dialog:

> And in some cases the name of the idea is not only attached to the idea in an eternal connection, but anything else which, not being the idea, exists only in the form of the idea, may also lay claim to it. I will try to make this clear by an example:- The odd number is always called by the name of odd?
>
> Very true.
>
> But is this the only thing which is called odd? Are there not other things which have their own name, and yet are called odd, because, althought not the same as oddness, they are never without oddness? [6]

[2] The term *form*, as used in this paper, means a single instance of a form, either blank or containing some data, which is moving through the system. There is only one template for each type of form, but there can be many different instances. A complete glossary of terms is found at the end of this paper.

[3] A *user* is someone who uses the form system just to fill out blank forms or use and forward existing forms. This is distinct from an *author*, who designs forms.

[4] A *stop* is a state in which a form can be, such as "pending supervisor approval."

Requirements of Phaedo, an electronic form routing system

Andy Oakland

June 16, 1992