FormStream a Workflow Prototyping Tool for Classroom Use

Paul Juell
and Benjamin Dischinger
Department of Computer Science
North Dakota State University
Computer Science
NDSU
Fargo, ND 58105
paul.juell@ndsu.edu

Abstract

FormStream is a tool for easily developing form based workflow systems. This tool allows students studying such systems and designers to concentrate on the overall system rather than details. It is easy to take an example form and a diagram of the states of processing the form and convert these into a working system. The tool is web based and produces web based form processing systems with a database backend. After building a system it can be directly run.

Introduction

This paper presents FormStream, a tool intended to aid students in learning about designing and building workflow systems. A typical workflow system tracks a form's process through many steps of processing. An example of a workflow system is an insurance claim processing center. A claim comes into the center and is sent through a number of standard steps to determine if the claim is to be honored. In the older paper versions of this process, a paper form would come into the center and would be handed from person to person to be processed. Each person processing the form would add information to the form and then send it to its next processing step. The current workflow systems support more complex options than the original paper system, but still have much in common with the original paper based systems.

There are systems available to build computer based workflow systems [Aalst 2002] but these are large scale systems designed to build a production system. The commercial versions are costly. Both the commercial and public domain systems have large sets of features and options to address fielding production systems. These tools were built to aid in producing high quality final systems. While this is an important goal, in general the same features interfered with rapid prototyping and seeing the overall properties of the system. That is, there were high learning curves and too many details visible making it hard to see larger issues. At the other end of the spectrum, we found numerous add-ons to a range of systems to support parts of the document flow process. The add-on systems typically were missing parts of the workflow model or did not have easy to use building tools.

Students Need Rapid Prototyping Tools

We wanted to teach students about designing and building workflow systems. While any one step of processing of a form is simple to show, with say a program, the overall process can not be seen from the small steps. We felt it was important for the student to see and understand the overall system. To make this system view possible we designed and built a rapid prototyping system. It was important that the system had a small learning curve and allowed building complete working systems.

Analysis and Design of a Workflow System

A typical use of a workflow development tool is to build an electronic version of a current paper form. In that case a data collection and analysis phase would be required. This would include going out and collecting copies of the forms used in the process, determining who processes the forms, the stages the processing goes through and what processing is done at each stage. At the end of this initial analysis stage you normally would have a flow diagram indicating the order of processing the forms and the various routes the form can take through the organization. The branches in the diagram show

various decision processes made by the organization. In addition, the procedure of interviewing the individuals performing the processing will give you the data entered into the form and the processing done at each stage of the process. After collecting all of this data, the analyst will evaluate the real needs and may restructure each form and the form's flow through the system.

After this analyses process, we have a diagram showing the flow of the forms through the system. Each node in the diagram specifies the current state of the form. The node then represents the processing to be done if the form is in that particular state. Included in the diagram are the branches identifying the routing to be done after the processing. The processing at a node is normally done by a particular individual or class of agents. For each state there will be a description of the processing to be done. This processing will normally be for a person to read certain fields and then write a value into another field. We will refer to the user group that does this processing as having a particular role in the process.

Nomenclature

In FormStream a complete package to process a form is called a system. When you create the system, you specify an attribute for each field in the form. You also specify the roles of the people processing the form. Each node in the diagram describes a state. The state describes the condition of the form before the processing. Each state has a default next state. If normal processing is not followed, a different next state can be specified. Each state can have a view specified. The view specifies the roles which can perform actions on the form and form presentation information. Once a system has been built you can execute it, by simply selecting the system and the desired role/view you want to process. A queue of forms waiting to be processed will be displayed. You can select a form, process it, and have the new values and state saved.

Specifying a System (Building the Data Base for a System)

You start the process of building a system in FormStream by selecting the action ‘create system’. You can then enter the fields found on your form. These are recorded as attributes in the system. For each attribute, you specify its name. All attributes are considered to be character, but you can specify the number of columns and rows for the display/entry box for the string. The attributes are basically the database description for each instance of a form. Figure 1 shows FormStream being used to create a Drop/Add form data base.

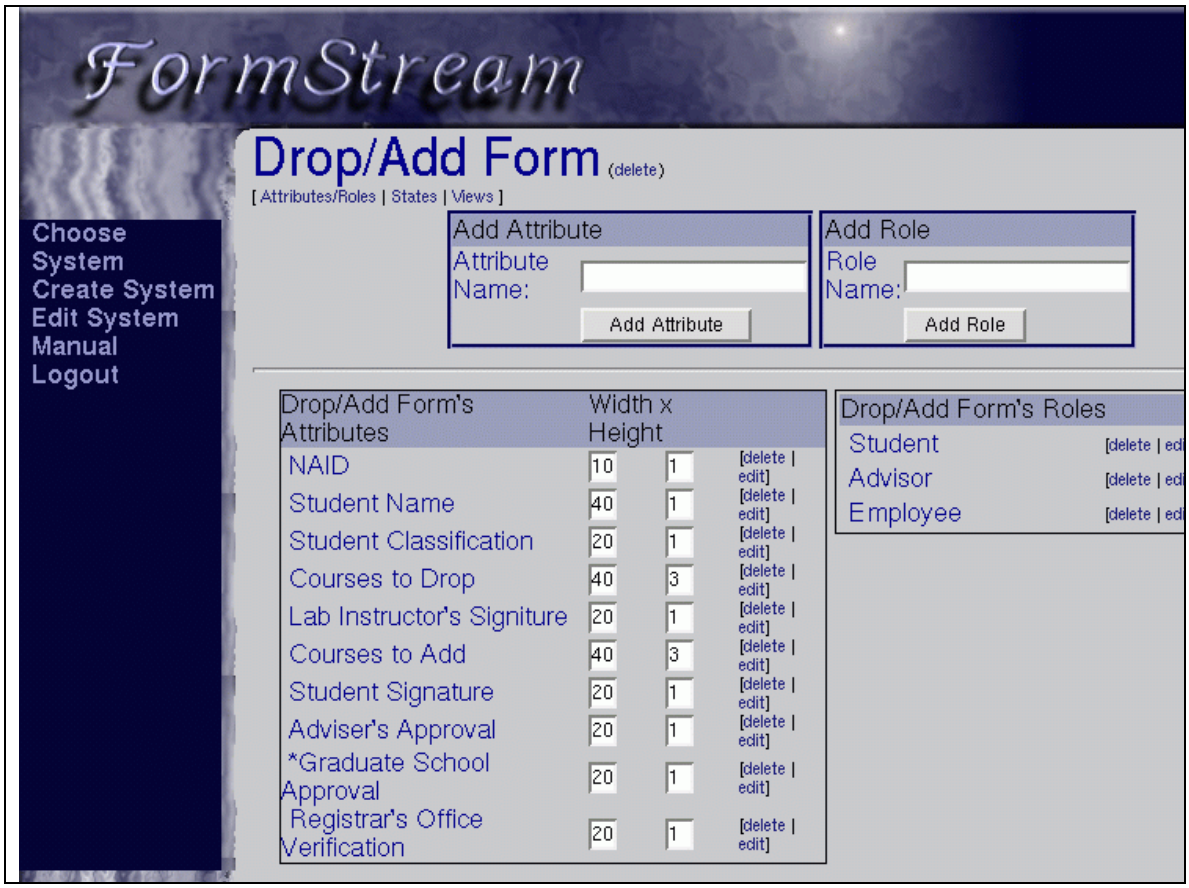

Figure 1: Edit attributes screen

Views and States of the System

After you have described the contents of the form, specify the states the form can be in. These states will be the names of nodes on your system flow diagram.

For each state specify the normal next state in the system. A special state is marked as the initial state, this is basically the starting blank form. We now have the normal flow lines of processing specified. Figure 2 shows the edit states screen. For each state, we expect an individual performing a particular role will process the form. So for each state, a view is specified. In the view is the state name, the role of the individual doing the processing, formatting information and information about data to be saved.

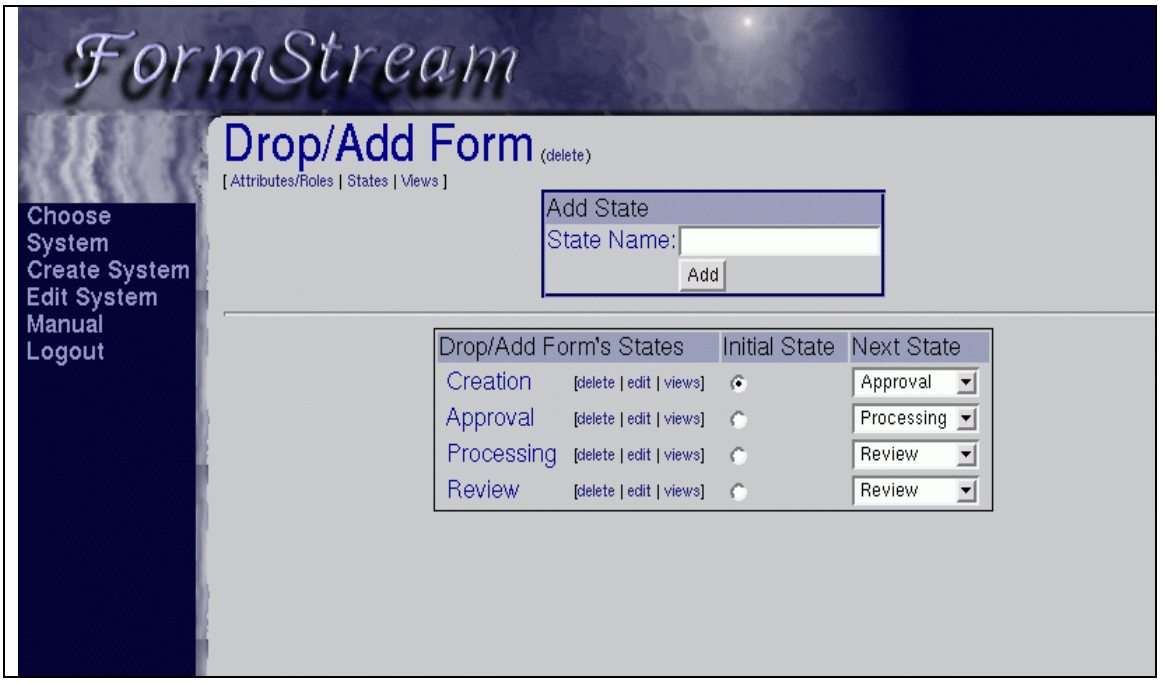

Figure 2: Edit states

Each attribute can be selected to be displayed or not for a view. If selected, you then choose if you are adding the data, editing the data or only displaying the values. Editable data will be displayed in a web form entry field. Not editable data will just be displayed. These choices allow simple but powerful tailoring of the display for the view. The choices can be used to give the effect of a related set of forms. In addition, detailed formatting control can be supplied. Each view has an entry box area where data formatted code can be specified and post processing code also can be specified. The base programming language is php and the code supplied in the view must also be in php. The formatting and post processing code, for that view is called by the page processing routine. These controls have been too used to control formatting, add text and instructions. The code is basically a subroutine with known variables for passing information in and out. A close analogue would be each of these units of code is plug-in for that view. Figure 3 shows the edit view screen with code supplied.

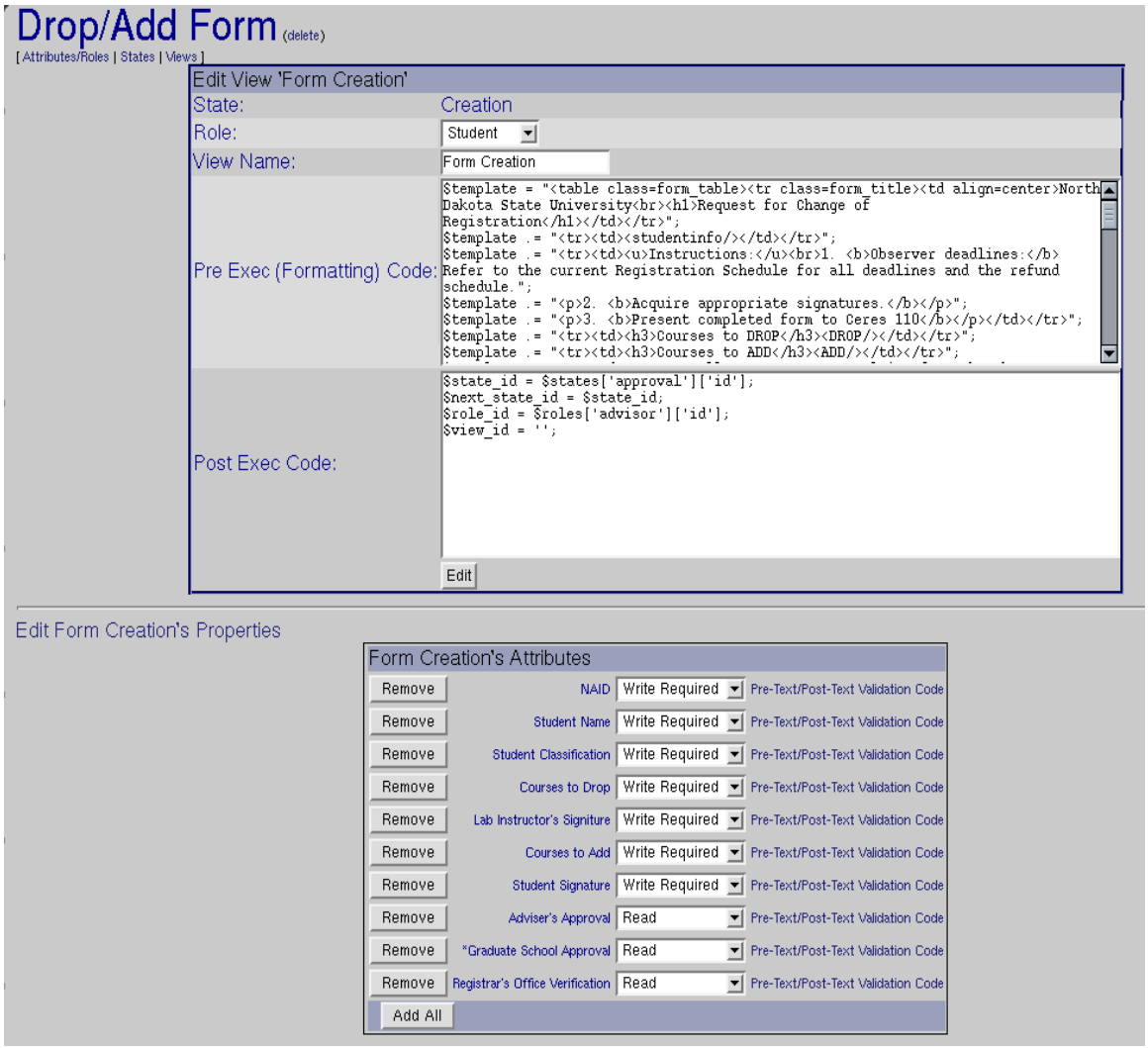

Figure 3: Edit view screen

Running the System or the User's view of Processing the Form

FormStream is designed to allow incomplete and rapidly developed systems to evolve into a mature system. Once parts of the data are entered, the system provides a working web based interface to the existing data. Initially, relatively simple default interfaces are provided. As the development process continues, the designer can augment the data base, the states and refine the views to be more attractive.

It was important to us to be able to quickly prototype systems and to be able to evaluate the overall system based on partial data.

Once the database has been described and some views are entered, you can run the system. All work starts by the user selecting the role and state for the initial blank form. This initial state allows the user to enter information and instantiates a copy of the form to be processed. Let us refer to all of the processing to be done on this copy of the form

as a job. The full job will take the copy of the form and send it through a number of states. Each state has a queue of pending jobs. For all states other than the initial state, after the user selects the role and state, he will be presented with the current list of jobs pending in that state. When a job is selected, that instantiation of the form will be displayed according to the current view. Figure 4 shows the initial state of a Drop/Add process with formatting provided by the code specified in the view.

Figure 4: Formatted initial state of a Drop/Add system

The simplest form of a view specifies which fields are to be displayed and which can be edited. The current copy of the form is pulled from the database and displayed. The selected items are displayed as text, on separate lines. The editable material is displayed in html editable form on the web page. When the user presses submit, the form fields are written back to the database with its new state indication. As the design process proceeds, more formatting and processing may be added to the view. Figure 5 shows a basic interface without additional formatting. It also shows the job queue for that state.

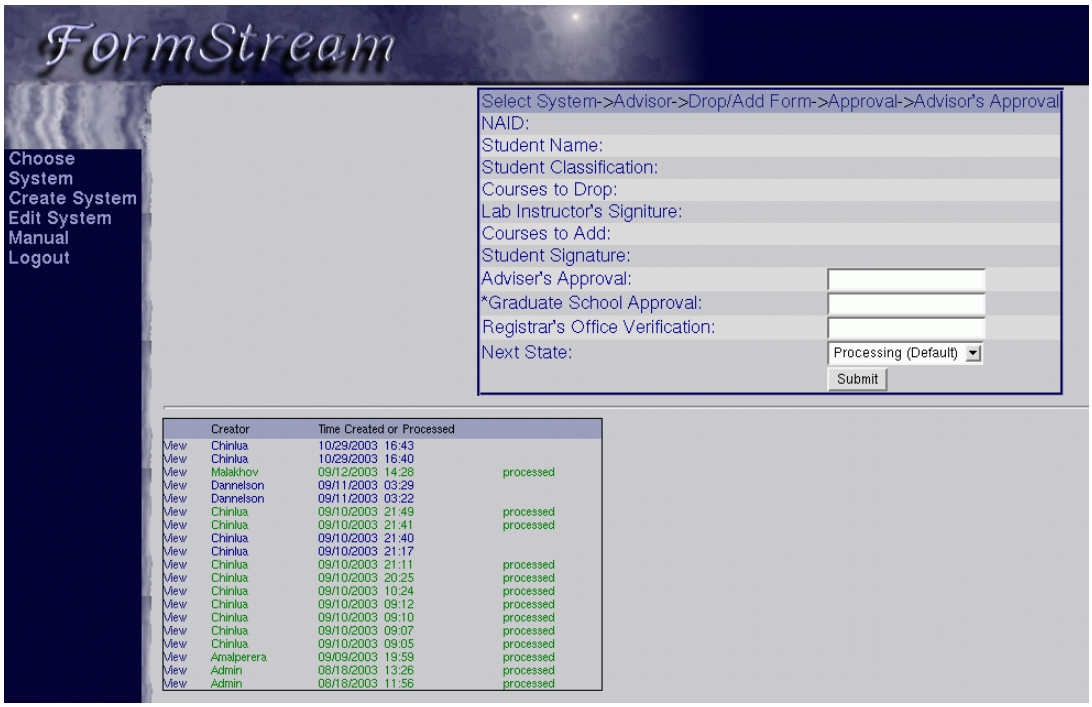

Figure 5: Another state of Drop/Add with with no formatting specified

The Internal Architecture

The tool was built in php and runs under the Apache server. The php program uses three sets of MySQL tables to store information. These sets of tables are for information about systems, the registered users and the jobs being processed for each system.

All of the information about the individual systems is stored in the database. This information includes the entries in the form, the states, and the views. The view description includes fields to display and edit. In addition, the view can have code to be run before the page is displayed and code to be run after the user submits the page for processing. This code is stored in the database.

The tool is intended to be used in an educational setting and by developers working together. It has a data base table of users and passwords. However, the tool has a philosophy of open enrollment. That is, you can easily enroll to the system. Once enrolled, you can act in any role handling jobs and editing the systems. This is reasonable for a limited access development environment. However, this would have to be restructured if the final versions of systems are to be used in a production setting.

There is a third set of tables for the queues of running jobs. Each system can be run. If you run the system, each instantiation of a form will create a data base record. This record includes all fields for the form, the current values and the instance's current state. The user interface will pull up job queues for requested states from the database. When an individual job is selected, that record is displayed by the view. When the user submits

the record, the optional additional processing is done and the data base entry is updated with the new values and the next state.

This basically means that after entering a limited description of the system, users can try the system. This allows early evaluation of look and feel and of functionality.

Strengths and Limitations

We used this system for a graduate level course. The course was NDSU's CS730, Office Information Systems, taught in the Fall of 2003. The students developed systems to mimic existing paper form systems in various areas. They worked in teams. All of the groups were successful in developing a working system, and had systems that looked like they would be useful in the problem domain.

Initially we had some stability problems, but over time, most or all of those were resolved. We think, that overall, we meet our goals of allowing the student learn about the system issues without spending extensive amounts of time on details.

A major limitation of the tool is that it does not deliver standalone systems. The students would have been happier and we would have better tools for developers if web pages for the job queues and views could be accessed through a system not including the development environment. This will not require much work on our part, because this always was the intent of the system. But it was not an issue we addressed in this first version.

Another limitation was the state engine. It would have been nice to have diagram interface to the state diagram. However, the lack of that did not seem to be real limiting factor on usability. The real problem was that we needed a better way to get from one state to another. As a first step, we should have had a restricted list of recommended next states. We had hoped that the ability to call code after user submitting a form would allow building a state engine. Unfortunately, we found the code somewhat hard to write based on the way we invoked the code. The system does work, but we want to look at better solutions addressing that part of the design. We think we can address the next state issues by adding another code section to directly select the next state from the restricted state list.

Overall, many things worked well. It was easy and quick to incrementally build up a system. The learning curve was very quick. Students could build complex systems and realistic formatting and control was provided. In addition, much of the system description was moved into a database. This allows clear description of the system and should allow reuse of the data in other contexts.

Summary

We have presented FormStream, a tool to allow students to rapidly develop prototypes of workflow processing of an electronic version of a form. The tools allow the student to easily evolve the prototypes into full systems with web interfaces and a backend database.

An important property is that the workflow system can be run in production mode at any time. This allows the student to alternate between design and evaluation.

References

van der Aalst, Will and Kiss van He,
Workflow Management:
models, methods and systems,
MIT press, 2002.