

The Development of a Scrip Management & Order System: A Case Study

Rod Sink

Department of Operations Management and Information Systems, College of Business, Northern Illinois University, De Kalb, IL 60115 USA
815-753-1287, rsink@niu.edu

Jack T. Marchewka

Department of Operations Management and Information Systems, College of Business, Northern Illinois University, De Kalb, IL 60115 USA
815-753-1332, jtm@niu.edu

ABSTRACT

This paper describes the development of a scrip management application system to support the fund-raising needs of a youth swim team. A scrip program is a common fund raising activity used by many not-for-profit organizations and offers a long-term method of raising funds without asking people to spend more money than they normally would in their daily routine. A scrip program requires that people purchase coupons, certificates or cards (scrip) from the not-for-profit organization or directly from a national scrip organization. The scrip cards or certificates can then be used as cash. Each merchant or vendor then contributes a set percentage back to the not-for-profit. Utilizing an action research approach, the organizational and technical decisions will be discussed as well as the lessons learned. This should be of interest to practitioners interested in developing Web-based applications, as well as researchers who hope to engage in action research.

INTRODUCTION

Youth sports teams often rely on fund-raising activities to support their existence and growth. To grow and remain a viable organization, a youth swim team located in the Midwest needed to find alternative sources of revenue. Subsequently, an innovative idea to develop a Web portal based upon a common fund-raising activity called scrip management was conceived and developed.

This paper provides a case study that describes the experiences and insight gained from the development of this application system. In general, a case study provides an appropriate research strategy when “how” or “why” questions are the primary interest, when the researcher does not have significant control over events, or when the phenomenon of interest takes place within some real-life context (Yin, 1994). In addition, descriptive or qualitative research may be undertaken when description and explanation are of greater interest than prediction (Merriam, 1988).

For this research, the primary focus is on describing and understanding the impetus for developing the application system, the decisions involved, the processes, approaches, tools, and techniques, as well as the lessons learned from the experience. As Mason (1996) points out, qualitative research should be constructed around an intellectual puzzle that attempts to produce an explanation grounded within the specific context of an applied problem.

In addition, an action research approach was followed in the development of the application system described in this paper. Action research is based on the work of Kurt Lewin (1973) and can be described as an applied approach to research. This alternative approach to traditional research focuses on solving real-world problems by simultaneously creating a change and then studying that change (Dick, 1999). This research may be considered action research since the primary researcher was an agent of change who participated actively in a collaborative effort to solve a real-world problem while acting both as an observer and interpreter.

The remainder of this paper is divided into four sections. The first section describes the background and impetus for the concept of developing a Web-based portal to support a not-for-profit swim team. The second section will describe the proposed business model, while the third section will describe the development process, the critical decisions that were made, and the system's features and functionality. The last section will provide a discussion of the lessons learned from the action researchers' experiences and how they may be applied to both practice and theory. Subsequently, the research reported in this paper should be of interest to both practitioners and information systems researchers.

BACKGROUND

The swim team currently has approximately 95 boys and girls between the ages of six and eighteen. It has been in existence since the early 1970's. Although the team is sponsored by a local YMCA, each child is required to pay a fee to swim on the team along with a \$25.00 YMCA membership fee. Similar to the wide array of ages, swimmers also vary in terms of their abilities and interest. The team supports beginner, intermediate, and highly competitive swimmers.

However, the membership fees only cover a portion of the team's operating costs. Additional pool time rental, salaries of the coaching staff, and other sundry expenses require that the team look to additional revenues. Although some revenues from swim meet concessions helps to some degree, the money earned still does not provide enough revenue to support the team fully. In addition, the concessions and other support activities during swim meets are largely supported by the volunteer work of the swimmer's parents. As is the case with most youth sports programs, the volunteer work is not always spread evenly whereby the team is consistently dependent on a small group of parents who tend to more involved than others.

The swim team has also attempted to organize several special fund-raising events. This includes the sale of such things as shirts or candy bars. Although the sale of such items has helped the team in the past, one drawback to this approach is that it requires the management of an inventory. Moreover, the sale of such items is often disproportional

among the swimmers or the parents of the swimmer often feel obligated to make a purchase. In such cases, the fund raising activity becomes equivalent to an additional fee.

DEVELOPING THE BUSINESS MODEL

In the fall of 2002, the president of the swim team and a fellow board member responsible for fund raising met informally to discuss and identify alternative sources of revenue to help support the team and allow it to grow. It was determined that the formulation of a possible solution would require no inventory and an adequate, ongoing stream of revenue that would not be an annual or semi-annual fund-raising activity.

The swim team president and the other board member in charge of fund raising both have a strong background in information technology. The swim team president, for example, is employed by a large, well-known software vendor, while the board member has worked and taught in the information system field for over twenty years. This led to discussion as to how they might leverage their experience and talent by designing and building an application system that would provide a useful service and would be profitable for the swim team.

After generating several ideas, it was decided that a scrip program would be the most promising. A scrip program is a common fund raising activity used by many not-for-profit organizations and offers a long-term method of raising funds without asking people to spend more money than they normally would in their daily routine. A scrip program requires that people purchase coupons, certificates or cards (scrip) from the not-for-profit organization or directly from a national scrip organization. The scrip cards or certificates can then be used as cash. Each merchant or vendor then contributes a set percentage back to the not-for-profit, ranging between 2% and 25%.

However, many of the existing scrip programs have manual order processing and order management processes. The idea was to automate these processes via the World Wide Web in order to provide a standardized interface for both customers and the merchants. The system would therefore act as a bridge service between customers and merchants/vendors and could garner revenues from both sides of this business model. It was believed initially that this project would be well-within the skills and resource requirements needed to develop and host such a system.

SYSTEM DESIGN

The system would ultimately need to address the functionality to service the customer and merchant roles of the business model, as well as system administration and not-for-profit organizations. Subsequently, this would require a set of system components that would include customer management, order creation/processing, perpetual inventory, and financial reporting. Figure 1 provides a Use Case Diagram that depicts the system boundary, the actors, the various use cases, and their relationships with the various actors.

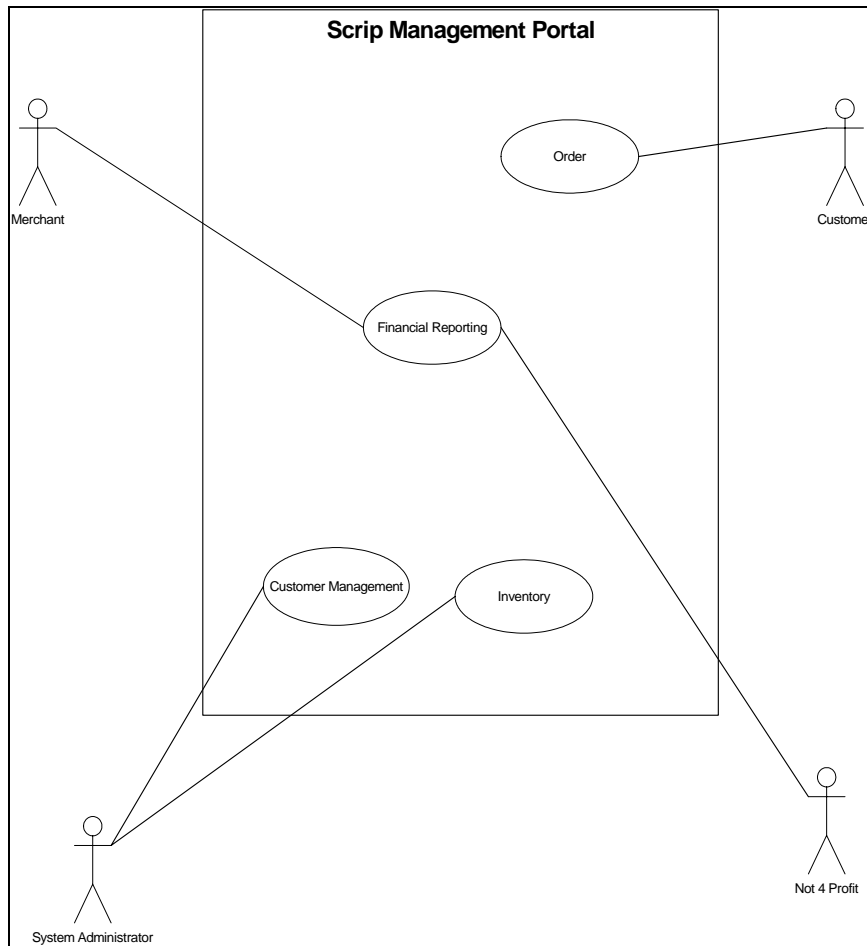


Figure 1: Use Case Diagram

Customer Management

The customer management component includes the management of customer data, customer security (i.e., login ids and password management), and database administration of the scrip vendor/customer hierarchies.

Order Creation/Processing

The creation/processing component includes the order summary display that is filtered by the login id a user provides when logging into the system. All users of the system use this entry point after login completion. Aside from providing the facility to create a new scrip order, this form also includes additional order display filters (e.g., backorders, open orders, sent orders, etc.) that provide the user with both summary and detailed information about past orders.

Inventory

A perpetual inventory component provides the functionality to manage and maintain a given merchant/vendor's scrip inventory. This includes the ability to manage the scrip available during order creation, reduction of script quantities after new order creation,

and additions to inventory as new scrip is received from a merchant or vendor. A future opportunity was identified whereby a supply chain component could be implemented to assist the scrip vendor through direct communication of inventory replenishment. However, this was not given a high priority at this time.

Financial Management

A financial management component can be integrated to feed or become part of an existing financial system. Many scrip vendors and merchants can be classified as small businesses with gross revenues of less than \$10 million a year. Subsequently, a commercially-available financial package is frequently used to manage the accounting functions. Although the financial data could be converted to an XML format, this type of integration may not be feasible with lower-end financial software packages. Therefore, a set of summary reports of the end-of-order cycle processing can be provided to the scrip merchants or vendors who would have to enter the financial information into their accounting and financial systems.

THE DEVELOPMENT PROCESS

Once the business model was conceptualized, the next step required making a decision regarding the technical platform to be used for development and deployment. Java/J2EE was chosen for its wide-spread industry penetration, stability, and platform independence. In addition, Sybase ASE, a widely-used relational database management system (DBMS), was chosen for its ability to integrate with the Java/J2EE platform and for its scalability. Moreover, the Java/J2EE and Sybase DBMS would provide appropriate technology tools for developing a portal and transaction processing system needed to support the business model.

It was felt that this approach would work out well for the board member who was responsible for the fund-raising activities. First, he would be the primary developer of the system, and this would provide him with a good opportunity because he had wanted to pursue a “meaty” project with one of the emerging J2EE tools for some time. Such an experience would provide him with additional professional experience as well as identifying a good Java tool that could be integrated with a course he teaches at a university.

However, before jumping too quickly into the development of the system, it was decided that a prototype or a “proof of concept” of the Web portal would be a wise choice. This would allow for test marketing the scrip management system to merchants and vendors without expending a great deal of time and resources upfront. However, a decision to use Visual Studio.Net’s VB-Winforms and Microsoft Access – a lower end DBMS – was made in order to develop the prototype quickly since the board member developing the prototype was already familiar with these tools. It was felt that the learning curve to develop the prototype using the Java/J2EE and Sybase ASE tools was too steep to develop a proof of concept in a short time.

Shortly thereafter, the analytical work to develop the prototype was started. This included defining a set of business rules based upon the multiple perspectives that the

system would have to handle. Once these rules were believed to be fairly complete, the next step focused on designing the logical and physical database design. Once the database design was normalized, work began on designing the user interface and business logic components that would be combined into a single systems component using VB-Winforms.

Several standard systems analysis and design practices were intentionally bypassed during the development of the prototype. This included any major front-end planning for a layered architecture (abstraction), security encryption, transactional security, or any overall performance concerns or efficiencies associated with writing the programming code.

Working on the application mainly on the weekends, the developer produced a prototype with some basic functionality within approximately two months. Functionality of the prototype included order entry, order display, order updating, order processing, inventory management, and role management to support the system administrator, not-for-profit, and the scrip merchant/vendor. The detailed attention paid to defining the business rules and fine-tuning the database design required only a few minor changes to the prototype when a function system review was conducted by the board president and board member/developer.

After the prototype completed its functional testing, it was decided that it would not be good enough to be demonstrated to potential customers. Too many VB-Winforms associated characteristics, such as certain functionality that only applies to client program execution and not a Web interface, lingered with the prototype and made it difficult to show how the system could be a viable Web-based application. A decision was made to move to a two-phased implementation of a production system.

The reorganization of the project was based upon the maturing of the marketing plan and very little to do with the technical, development work. The original project was framed around business functionality of customer management, order entry/processing, perpetual inventory management, and integration into existing financial systems. Subsequently, phase one would be limited to the customer management and order entry/processing components. This aligned with the revised marketing plan to pilot the scrip management portal to only the swim team's families. This would allow for more control with respect to any issues or problems that might surface with a new system. Once confidence in the system's stability was gained, the marketing plan would call for the system to be offered to other not-for-profit swim team organizations.

Phase two would focus on implanting the functionality needed to support the scrip merchants. This would require the completion of the customer management, perpetual inventory, and financial systems integration components. By placing these components in the second phase, more time could be devoted to firming up the marketing plan for the user/role of the script merchant or vendor.

At this time, the president and board member began searching for a Java-based development tool. Though several well-proven tools were considered (e.g., WebSphere

and JBuilder), it was determined that these tools would be outside the swim team's available funds.

As a result, it was decided that Visual Studio.Net's Webforms and the ASP.Net framework could be used to develop the required Web-based application and could be acquired within the funds available. The developer then converted the working VB-Winform prototype into VB-Webforms with the intention that a DBMS would be decided upon later. However, this somewhat determined that the ultimate application would be developed within a Microsoft environment.

After two months of part-time work phase one was complete. This included converting the existing prototype to VB-Webforms. The developer was pleasantly surprised to find that a large portion of the code developed with the VB-Winform was unchanged when imported into the VB-Webforms version of the prototype.

LESSONS LEARNED

Although the script management portal is still under development, there are still several lessons learned from this project. These lessons learned can be useful not only to practitioners and researchers, but also to instructors who teach information system courses.

First, project management, systems analysis, systems design, and system development approaches do not necessarily change whether the target development tools are the perceived "older-tools" or the "newer-tools" of today. It seems that the more "things" that have been perceived to "change," the more they may need to remain the same. This appears to be evidenced from the experience of the developer who approached this project with the same proven methodologies and approaches that he, as a project manager and developer, used for years. The system came to fruition with a minimal number of design issues. Often many inexperienced developers and students perceive that the "tools" alone will make them skilled analyst/programmer when, in reality, the tool is just that...a tool... that can be only effective in the hands of a skilled and knowledgeable professional. There is no substitution for efficient and effective planning, analysis, and design.

Second, the same technical issues that were being addressed over 20 years ago with pseudo-conversational transaction delivery systems, are the same issues that Web transaction development/delivery tools must deal with today. Even though we may be working within a Web paradigm, the only thing that may be new under this paradigm is the presentation layer and the tools to address that component of a transaction processing system. Many of the technologies that are needed and are used to develop and deliver transaction-based systems are now just being "discovered" by the "techies" of today. Their need has been well-documented for many years and their presence well-known for just as long. For example, this includes transaction processing monitors, multi-threaded applications, shared database threading, user/program context between iterations, and minimized connection thread use. Though this project will ultimately include these technologies, they are minimally addressed by the .Net tools. It appears that the old is still new.

Third, during the search for Java/J2EE development tools all of the tools “supported” transaction-base development through frameworks (e.g. Struts) but presented relatively weak support when compared to the focus and integration that the .Net platform inherently provides. Java/J2EE tools typically provide multiple ways to accomplish a give task, (e.g. relational database connectivity). Although this provides more latitude and flexibility, it requires the developer to make many front-end decisions and to “know the impact” of those decisions beforehand. This is often not the case with a professional who is new to a tool or with the inexperienced developers we have in the classroom setting.

Fourth, the component architecture is still alive and well. This may be a very good thing. Many industry professionals and educators tend to believe that if they are using a development tool that is object-based, they are designing and developing systems with an object-oriented approach. Unfortunately, this is not the case. In fact, the observation can be made that the .Net development tool actually encourages the construction of components instead of object-oriented applications. This definitely can be observed with Visual Studio-Winforms and is evident with Visual Studio-Webforms because the emphasis tends to be on development of the user interface. Though it is possible to build applications with component architectures in Java, it is not the direction of Java and Java development guidelines. This may sound a bit negative towards the .Net tool, but in reality it is not. Some have argued that pure object-oriented design and development philosophies have been over-sold since their inception. They fit and work fine as long as you know when to choose them for a project. Functional business transaction systems may not be the proper fit and this project provides some evidence. In short, one should match the development approach to the project.

Fifth, it is nice to live the lessons that we try to teach our students – i.e., the “business need” must proceed the technology-infused solution. Too often students and inexperienced developers believe that the technology is the most important driver in systems development. Technology is only a tool or an enabler.

Finally, a seasoned "old-technology" (e.g., CICS or a related TP monitor-based development tool) professional can become proficient with VS-Webforms in a relative short period of time. However, this may not be a drastic change for the developer. A majority of the transaction-based experiences and skills that these people possess probably do not have to be "re-tuned". Many of the same issues that these professionals have dealt with are the same issues that must be addressed in the ASP.Net executions environment. For example, this includes pseudo-conversational iterations, handling first and subsequent iteration processing, session state preservation, and maintaining database data needed between iterations either on the form/screen or when placed into a session state. These concepts cover a large portion of past transaction processing training. One major “change” that these folks may need to make is to substitute program switches (for logic/coding execution) with event driven programming, which is a relatively easy concept to grasp. The Web and the development tools of today do not necessarily change drastically the way that a transaction professional needs to think when conducting analysis, design or programming under the current Web paradigm.

REFERENCES

- Dick, B. (1999). What is action research?, <http://www.scu.edu.au/schools/gcm/ar/whatisar.html>.
- Lewin, K. (1973). Action research and the minority problems. in *Resolving Social Conflicts: Selected Papers on Group Dynamics*. (ed. G. Lewin). London: Souvenir Press,197-211.
- Mason, J. (1996). *Qualitative Researching*. London: Sage Publications.
- Merriam, S.B. (1988). *Case Study Research in Education: A Qualitative Approach*. . San Francisco: Jossey- Bass, Inc.
- Yin, R.K. (1994). *Case study research: Design and methods*. Second Edition. Thousand Oa