
Museum Information Kiosk Evaluation

Project Review

digisoln.com

Table of Contents

Overview	3
Project Requirements	3
Client Objectives	3
System Functionality.....	4
Client Expectations	6
Client Reflections	7
Work Breakdown	7
Scheduling.....	9
Risks	11
Reflections	12
Planning and Implementation of the User Interface	12
Microsoft Access 2003 as a Multi-user DBMS	12
SQL Injection Vulnerability.....	13
Location Information Stored in XML.....	13
Upload Images	13
Use of Automated Coding in Administration Forms – “Wizards”	13
Controls, Objects and Processes to Enhance UI Experience.....	14
Errors at Run-time.....	14
Debugging Environment Updates.....	17
Add New Buildings	17
Auditing XYZZY	17
Future Directions	18
References	19

Overview

XYZZY Software internally funded development of a proof of concept system for an information kiosk for the Whoop-Whoop Automotive Museum. The system will be demonstrated to the Whoop-Whoop Automotive Museum on Friday, February 4th 2010. Following completed work on the system, XYZZY's quality assurance requires a project review. The following is an evaluation of both the product delivery and development processes employed within the Whoop-Whoop Automotive Museum Information Kiosk project. This includes a discussion of the risk and management schedule from the engineers' perspective.

Project Requirements

The initial requirements of the project were delivered to the client in the Project Planning documentation, and these were further synthesised in the Requirements Specification. To best determine whether or not the projected targets were met, it is necessary to first measure the final product against both the initial client objectives and required system functionality. The results of this will determine whether client expectations were met or not. Finally, client expectations and satisfaction will be reflected upon.

Client Objectives

Client objectives were initially detailed in the project planning documentation, which was the first milestone achieved in the development process. The following results outline the product success testing mapped against each one of these objectives:

Objective	Result	Justification
<i>Display information on:</i>		
Current exhibits (including detailed information on items within a collection);	<input checked="" type="checkbox"/>	Visitors can browse and search items and collections, as well as view detailed information about both items and collections if requested.
Exhibits on loan (to other museums);	<input checked="" type="checkbox"/>	Visitors can view items on loans to other museums, including type of loan (incoming or outgoing), the location of the borrower / renter and the duration of loan.
Museum facilities (e.g. cafe, toilets, exits etc).	<input checked="" type="checkbox"/>	Available through the search facility. Each item and collection explains its location. Current location can be found in the directions section.
Calculate the shortest distance between two given areas within the museum;	<input checked="" type="checkbox"/>	The directions section uses the breadth first search algorithm from the Design Documentation to return the quickest path between two locations.

Be proven to perform reliably without error through thorough and documented testing ;	<input checked="" type="checkbox"/>	Quality assurance at XYZZY requires a fully tested system. As yet, the system has been alpha tested by <i>independent users</i> within the engineers' site. Beta testing, involving a limited public audience of museum visitors outside of the internal acceptance testing at XYZZY has yet to occur.
Be published with a set of user instructions;	<input checked="" type="checkbox"/>	A brief set of user instructions as required by quality assurance has been produced. This involves both visitor and administrator instructions.
Be used to further facilitate the review of XYZZY Software processes, procedures and organisational structures in an attempt to gain XYZZY Software ISO 9001 accreditation.	<input checked="" type="checkbox"/>	The project was successfully utilised for an internal review of XYZZY processes, procedures and structures. This project allowed the gaps to be identified for aid XYZZY in gaining ISO 9001 accreditation.

It is clear from this analysis that client requirements were mostly met. However, the lack of sufficient beta testing remains an issue that requires attention. To avoid unnatural or unexpected results at proof of concept demonstration, it is recommended that the software be released to an *uncontrolled* yet monitored test group, in order to increase the significance population that is being tested – and thus decrease the incidence of undiscovered bugs being allowed through to distribution.

System Functionality

The Requirements Specification document outlined specific system functionality and user case studies illustrating this, which was passed as a milestone and thus delivered to the client as their expected benchmark. Given this was agreed upon before development, it is another useful measure of client satisfaction in delivering what was promised:

Functionality	Result	Justification
<i>Visitor:</i>		
Exhibition Information		
Reliable, up to date information on specific collections held;	<input checked="" type="checkbox"/>	Information is maintained by industry professionals using the administration facility.
Detailing of iconic exhibits within collections and as standalones;	<input checked="" type="checkbox"/>	All exhibits and collections allow both brief and full descriptive details if requested.

Promotion of temporary exhibits (including future temporary exhibits);	<input checked="" type="checkbox"/>	Featured items are displayed; however, as yet, there is currently no provision to display information regarding future exhibits. This was deemed outside the scope of this project.
Exhibits on loan, their duration of loan and location of loan;	<input checked="" type="checkbox"/>	Loans feature allows visitors to view this information.
Building location of all exhibits held in the Whoop-Whoop Automotive Museum.	<input checked="" type="checkbox"/>	Buildings locations (and their exits) are stored dynamically and can be updated by the administrator.
Navigation of Museum		
Listing of all locations in the museum, including buildings 1 to 12, gift shop, cafe, toilets and both exits;	<input checked="" type="checkbox"/>	Locations are fully listed in a relational table that can be modified by the administrator.
Based on visitor selection, provide a simple, easy to follow set of instructions to reach their chosen destination.	<input checked="" type="checkbox"/>	List of buildings to pass through is located next to the location selection box. This list is naturally traversed in order down the page by the visitor.
Efficient Interface		
The kiosk must cater for a variety of visitors, include children, aged, and public with accessibility issues or other special needs;	<input checked="" type="checkbox"/>	Larger than standard system font, manageable tool panels and a search facility have been added to cater for all levels of users. Help is available throughout use of the software.
As such, the interface must be simple, intuitive and interactive, with a correct yet simple use of language, large fonts, and a consistent navigation structure with on-screen help wherever possible.	<input checked="" type="checkbox"/>	Navigation structure is consistent with top menu elements listed inline across the header, and work panels changing dynamically underneath. Pop up forms are displayed in modal dialog, so that the naturally progression of forms is not compromised (visitors must click back – they cannot jump forms).
<i>Administrator:</i>		
Update Exhibition Information		
Maintain correct descriptions and historical information of exhibit holdings;	<input checked="" type="checkbox"/>	Administrators are designated by the museum and given access rights; these historical professionals are accountable for the correct display of information. The software allows for

		access for these professionals to do this, without the public being able to access this facility.
Add new exhibits to the system, remove old or unused exhibits, and modify or moderate content in existing exhibits;	<input checked="" type="checkbox"/>	New exhibits can be added to the system using the Administrator facility – this information can be modified as the Administration sees fit.
Set locations of exhibits, based on the building layout supplied;	<input checked="" type="checkbox"/>	Building locations can be adjusted using the Administration pages. Further to this, the actual building layouts can be adjusted dynamically using the same facility.
Record and track on-loan exhibits, including date of loan, client information and return date of exhibits;	<input checked="" type="checkbox"/>	Full loan logging facility with client detail storage available.
Modify, add or remove promotional, temporary or advertised exhibit content;	<input checked="" type="checkbox"/>	Featured items can be made features by using the make feature column; technically this uses a 1 for true and 0 for false, which allows items to be displayed as promotional feature items.
Set and change the current kiosk location.	<input checked="" type="checkbox"/>	Implemented by use of an XML file which can be adjusted in the Administrator pages.

Evidently, this shows nearly all of the required functionalities were delivered. In addition to this, additional processes were added to realise best practise for the museum; including:

- a search facility;
- the ability to change the layout of the museum and buildings (in case of extensions or alterations occur);
- the ability to change administrator username and password.

Given the above results, client expectations and satisfaction can now be reflected upon.

Client Expectations

One of the seven objectives of the project was not fully delivered; this was the resultant beta testing, which delivers a significant risk to quality assurance to the product. In light of this, given that both the scope of the product and that alpha testing was substantially completed, a relaxed opinion may be taken in the weighting of this risk from a client point of view. In regards to system functionality, the inability to promote *future* temporary exhibits is unfavourable, as this reduces the exposure and reach the kiosk has in attracting repeat business. If the client receives the kiosk as a tool to provide *current* exhibit information, and

other avenues of advertising are utilised to promote future exhibits, then in this instance the expectation may be still be favourable.

Given this, and the cost of production (discussed further below), the satisfaction measurement of the client is expected to meet approval of the product. It is important to note here that continued support to the lifecycle of the application must be provided in agreement to maintain this level of customer satisfaction, improve reputation and better ensure repeat business.

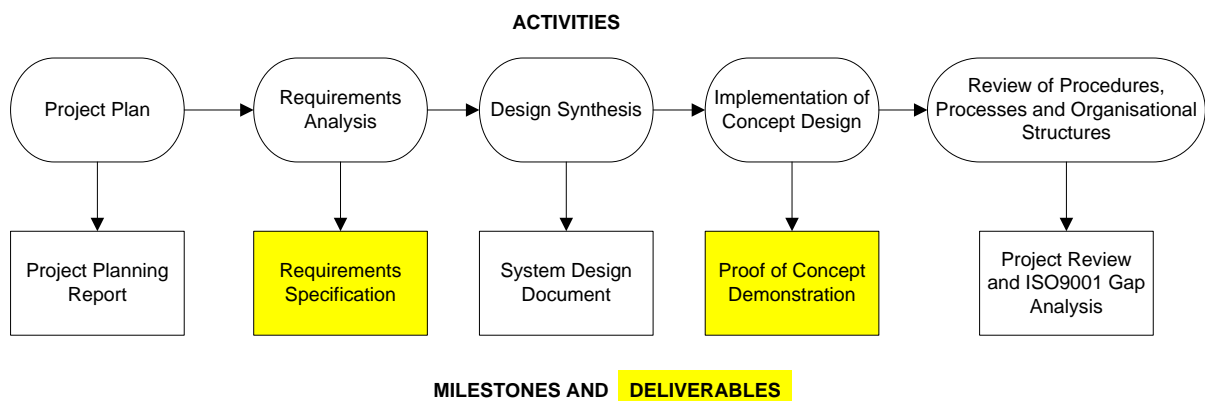
Client Reflections

The engineers – asked to reflect on client expectations – believe that the client expectations are always reasonable if the quality assurance process ensures an agreement between both parties for a required product to be produced. In this case, the client expectations were outlined in detail in pre-production documentation. The engineers managed risks, and built extra time into the development cycle to best meet these requirements. Agreed upon tasks were not beyond the ability or the belief of the engineers regarding both the feasibility and engineering requirements; any perceived or un-encountered risks were mitigated, with further steps in place (that were not needed) to ensure success (e.g. outsourcing).

The critical end result is that client satisfaction was delivered. This was identified as critical by the ISO 9001:2008 gap analysis, of which the results (of this analysis) must be resolved into affirmative action for better quality processes and management. The focus of this, and the processes undertaken during this project, were all geared towards meeting client satisfaction and ensuring quality control. Given that the client expectations were predominantly met within the scope of this project, both under budget and within time and resource constraints, the client consultation throughout this process has been effective.

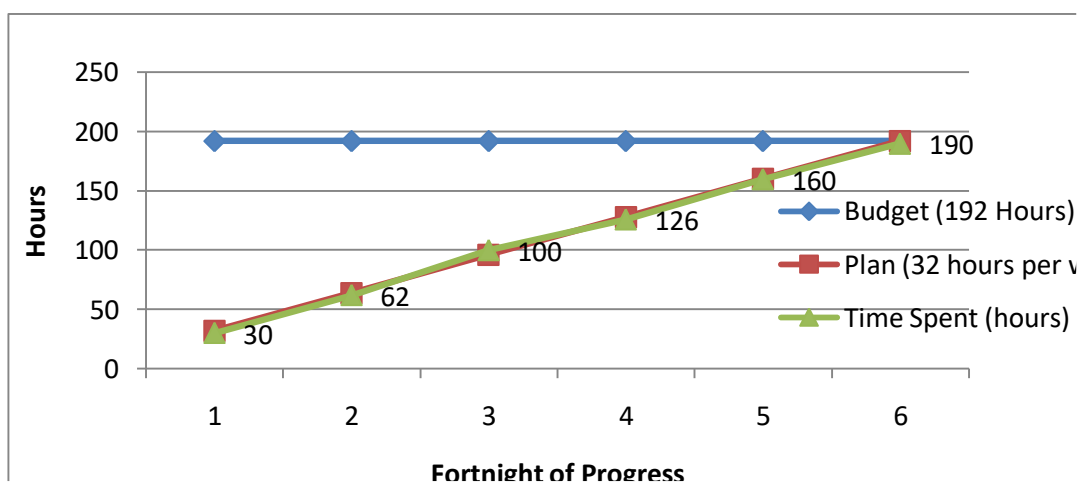
Work Breakdown

The work breakdown structure employed throughout this process was determined in the initial Project Planning documentation. The following diagram was included in this initial planning document, which shows the breakdown of activities, their associated milestones and highlighted deliverables to the client:



This work breakdown effectively managed the twelve week development process into activities, milestones and deliverables that could more readily be allocated time and resources. Each of the milestones and deliverables enabled clear client and engineer communication, and further to this offered another indicator of project performance to date (in addition to the fortnightly progress reports). In this capacity the work breakdown for this project was effective and served its necessary (and intended) function.

The fortnightly risk reports evaluated progress and provided measurable data on resource and time consumption. From the development graph of time as shown below, it is clear that in week six (17th December 2011 – **fortnight 3** in the graph below) development was ahead of time. This fortunately counter-acted the week eight (7th January 2011 – **fortnight 4** in the graph below) Rockhampton local flooding which caused significant time delay, and of which the resulting project development *for that fortnight* fell behind schedule.



The above graph further indicates that development was completed **two hours** under budget. Given the budget was 192 hours, to complete this project in 190 hours can be seen as a negligible difference; however, to have *any* surplus of hours in a project budget is the ideal cost effective situation for XYZZY, as it means no further costs are incurred. In a different situation, a time / resource deficit may have resulted in further financial costs (such as outsourcing), or an incomplete product that is more likely to fail quality assurance.

Given the above discussion, it is the authors' opinion that the budget constraints allowed by XYZZY were reasonable for the agreed client objectives and system functionality. The budget was also **proven flexible** as it allowed mitigation strategies to be put in place for unexpected events (e.g. Rockhampton floods). The budget successfully met the requirements of the task, with neither over expenditure nor over estimating resources causing any resulting loss to the productivity of XYZZY. Finally, it can be also concluded that not only did the budget *fittingly* meet the requirements of the task *on review*; it also met the requirements of the project *throughout development* to an acceptable standard for engineers to perform competent work; this was further reflected in the budget assessments for each fortnightly progress report (green line above), with little deviation from the planned budget (red line above).

The work breakdown offered a detailed breakdown of activities, milestones and deliverables, which enabled effective, efficient and further breakdown of tasks in project scheduling. This aspect of the development process will now be investigated.

Scheduling

The project time schedule was delineated in the initial Project Planning documentation, the foundation of which was the project dependencies (also outlined in this initial document). Inherently, the project dependencies were based on the milestones discussed above in the work breakdown section. Thus, before investigating the time scheduling for the Whoop Whoop Automotive Museum Information Kiosk project, it is necessary to investigate the foundation dates to which milestones were reached.

The following table illustrates the overall completed milestones (and deliverables) with their week (and percentage) completion, taken from the work breakdown (detailed above). Matching this with the completed project schedule, it is clear that *all six milestones successfully align with their allotted project time scheduling.*

Milestone	% Completion	Description	Week Completed
1	100 ✓	Project Planning Report	3 (19 Nov 2010)
2	100 ✓	Requirements Specification	5 (3 Dec 2010)
3	100 ✓	System Design Document	6 (14 Dec 2010)
4	100 ✓	Proof of Concept Demonstration	10 (21 Dec 2011)
5	100 ✓	Project Review	12 (4 Feb 2011)
6	100 ✓	ISO 9001 Gap Analysis	12 (4 Feb 2011)

The results in the above table suggest that project dependencies were successfully fulfilled to complete work on tasks with these pre-requisites. To successfully ascertain if this is the case, it is necessary to look further at the actual project dependencies listed, in order to better evidence the final measurement of the project scheduling:

Task	Dependent Task(s)	Assessment
Requirements Analysis and Design Synthesis	Approval of Project Plan	As per the fortnightly report for fortnight two (dated 26 th November 2010) the project plan was completed in week three. This was on par with the project schedule, and thus successfully met this dependency.
Database Implementation	Database Design (in System Design Document Milestone)	According to the fortnightly progress report for fortnight three (dated 17 th December 2010), these dependent tasks (the <i>Requirements Specification</i> and <i>System Design</i>) were both completed in this fortnight, as per project schedule, allowing the tasks with prerequisites to commence. Furthermore, it can be noted that in this
Application Layer Implementation	Class Design (in System Design Document Milestone)	
Visitor / Object Layer Interface	User Interface Design – Visitor (in System Design Document)	

	Milestone), Use Case Scenario – Visitor (in Requirements Specification Milestone)	fortnight, a total of 34 hours were worked by the two engineers which was <i>two hours over planned working time</i> – reflecting the greater perceived need to complete these dependent tasks . The project scheduling was still correct despite the extra time worked; through doing this extra time, the engineers created a budget surplus of time, which allowed more time for future risk mitigation strategies (as eluded to previously, the floods); as well as the early completion of tasks (in this instance).
Administrator / Database Interface	User Interface Design – Administrator (in System Design Document Milestone), Use Case Scenario – Administrator (in Requirements Specification Milestone)	
Review of Procedures, Processes and Organisational Structures	Proof of Concept Demonstration Milestone	The programming was completed well in advance of both the ISO 9001 gap analysis and the Project Review, as per project scheduling requirements.

Given that the milestone and dependent tasks had been met (demonstrated above), the project schedule ultimately (accurately) scheduled required tasks within the allotted time frame on *review*. During development, as has been discussed above, the relationship between lost project time (*specifically fortnight four – the Rockhampton floods*) and actual task scheduling is evident. In this fortnight, the scheduled tasks and their resulting loss of time (and counter measures) are shown in the following table:

Modification of Scheduled Tasks for Fortnight Four in Sequential Order for Each Developer – Rockhampton Floods:

<i>Engineer A:</i>	
Application Layer Implementation	Started early in the fortnight due to the previous fortnights surplus. No change to time production but task completed by 26 th December (change to project schedule).
Visitor / Object Layer Interface	Lost unexpected task schedule time 4 th – 8 th January. Visitor / Object Layer Interface started 31 December and completed by 4 th January – still 5 days development but an adjustment from project schedule.
<i>Engineer B:</i>	
Database Implementation	Started early in the fortnight due to the previous fortnights surplus. No change to time production but task completed by 28 th December (change to project schedule).

Application Layer / Database Interface	Lost unexpected task schedule time 3 rd – 9 th January. Application Layer / Database Interface started 29 th December and completed by 2 nd January – again, 5 days development but an adjustment from project schedule.
--	--

Thus, the project was *not consistently on schedule* (as demonstrated above), but was *consistently ahead of schedule* due to the surplus carryover of hours. Given this, it can be unequivocally agreed upon the project schedule was reasonable for both engineers, as has been proven by the successful scheduling demonstrated above – no evidence exists to the contrary. Finally, a relationship exists between the unsatisfied requirements of the system (above) and the project scheduling – the project schedule (located in A3 paper size in Appendix A of the Project Planning) does not make provisions for **recognised beta testing**. Alpha testing commenced as an action-ongoing process throughout the coding of the Information Kiosk; as such, this was not formally recognised as a task in the project schedule. Both of these are necessary steps which must be acknowledged in future project scheduling, with an appropriate proportion of time allocated to each.

Risks

A very detailed risk plan was introduced in the Project Planning documentation, and this was monitored throughout the development of the project in the Fortnightly Progress Report. The risk management system implemented a formula to give a quantitative assessment of the dangers presented; this formula was *Jeopardy = Probability x Effects*. The probability was measured as Low (0.25), Moderate (0.5) and High (0.75), with the effects measured as Insignificant (1), Tolerable (4), Serious (7) and Catastrophic (10). Although some risks were measured as High (e.g. an incomplete product due to time delays during the floods), as well as Catastrophic (e.g. engineer incompetence pre-production), neither combination of the two ever resulted in a jeopardy value equal to 6 or greater – which was the “magic number” where it was decided alternate affirmative action or measures needed to be implemented. The closest to jeopardy came early in development – in both fortnight two and four, where the jeopardy of an incomplete product risk totalled 5.25. It must be noted here that this was its initial risk assessment; although it remained constant for 5 weeks, due to increased production in fortnight three this risk was downgraded, and never again reached above its initial total.

In effect, the above example shows that risk management occurred within a controlled and documented framework; the success of which can be attributed to the quality process requiring Fortnightly Progress Reports (as outlined in the initial monitoring and reporting mechanisms). Notably, in week 4 (fortnight 2) the engineers’ identified the incumbent risks as unmanageable due to the sheer number; and as such a potential source of lost time. This was negated later in the project (as illustrated in the progress reports) by both the successful surplus of time in week 6, and by completing milestones allowing start-up risks to be negated (which lowered the number of risks to be managed).

Finally, in week 12 (fortnight 6) risks still remain regarding future challenges in both competition and on-going support of hardware; the engineers’ recommend further consultation with XYZZY and the client to reach agreements for warranty and length of technical support / updates to continue. The

monitoring of mainly project and product risks was key in delivering a quality system. Rarely were **business risks** compromised; these included both technology change and competitor kiosks. This is unsurprising; given the domain and scope in which the engineers worked, it was unlikely that supply of software development competitors in this field at this level would out-compete the XYZZY project; furthermore, the technology would have unlikely changed in twelve weeks. At the end of the ten week cycle – and continuing into the twelve week cycle, it was also noted that risks did not need to be further monitored, as completion of work had negated the risk (i.e. Jeopardy value of 0). This made management of the risks in the final weeks of production (i.e. the “crunch” end) more feasible and especially welcome.

Reflections

The above documentation has detailed the concerns, issues and risks of the Whoop Whoop Information Kiosk project management. Predominantly the quality assurance process of the combined documentation has reviewed and suggested recommendations throughout the project to achieve a greater degree of transparency in management. Remaining issues not covered above regarding the full development of the software will be discussed in this section, before (finally) the project can be concluded by discussing future directions for the engineers based on these lessons learnt.

Planning and Implementation of the User Interface

As the final build shows, the aesthetics of the interface layout planned differed based on the Visual Studio 2005 IDE. The engineers found themselves limited by (or rather to work within) the Windows Common Controls available. The work flow panel system implemented was identified by the alpha testing as a preferable method of control (to the collapsible menus shown in the project plan). This actually produced an easier to navigate system, as form clutter could be decreased (only 1 window panel would be shown at once, as opposed to the Search / Browse / Directions etc. all being displayed).

Although this forced change may have been beneficial, it was nonetheless a deviation from the detailed UI agreed upon by the client; in hindsight, a proposed change should have been formally made in writing, submitted to the client and accepted for implementation – otherwise the risk (currently) of a non-conforming product has been increased.

Microsoft Access 2003 as a Multi-user DBMS

Traditionally, Access has been known as a non-industrial or non-robust database management system when the number of users exceeds the number that may be found in a small application pool (e.g. twenty users). As specified by XYZZY, Access 2003 was to be utilised to complete this project. Should this project grow, and offer web or mobile based access to the data storage in a similar format as the current information system, inherently the current database management system may need review – and perhaps changing to Oracle or SQL Server.

SQL Injection Vulnerability

Currently, the application interface has left the database open to SQL injection attacks through current use of **concatenated strings** for SQL queries. For example, the following query appears in the information kiosk which is *string concatenated*:

```
"SELECT BuildingName FROM Building WHERE BuildingCode = '"+code-  
>Trim()+"'";
```

This should be rewritten using something similar to:

```
"SELECT BuildingName FROM Building WHERE BuildingCode = @code";
```

Which would be stored in a object (e.g. "querystring") that could employ use of the `System::Data::SqlClient::SqlCommand` object to add parameters, e.g:

```
querystring->Parameters-  
>Add(SqlClient::SqlParameter("@code", SqlDbType::NVarChar, 30)
```

This would result in much safer code being executed, and better assure ongoing integrity of the completed product. An example of this can be found in the **adminLocationsUsers** form, which utilises MD5 encryption for secure storage of the Administrator password:

```
cmdPassword->Parameters->Add(gcnew  
OleDbParameter("username", OleDbType::VarChar) );  
cmdPassword->Parameters[0]->Value = this->txtNewUsername->Text;  
cmdPassword->Parameters->Add(gcnew  
OleDbParameter("password", OleDbType::VarChar) );  
cmdPassword->Parameters[1]->Value = passHash;
```

Location Information Stored in XML

Currently, location information is stored in Settings.xml. This offers remote and alternate access by various middleware applications (as it was intended for future development and legacy changes); however it is believed this file will be stored in a secure location with restricted access. However, upon reflection, a more feasible solution may have been to simply store this information in the database itself.

Upload Images

Currently, no facility exists for file upload of images for the museum. Administration must copy the images manually to their respective folder, and enter the image link in the database itself (as text). Provision for an upload mechanism must be required to satisfy full autonomy from other middleware applications for administrator use.

Use of Automated Coding in Administration Forms – “Wizards”

The main customer interface was coded without the use of automated features offered in VS 2005. This gave the developers greater control over the access and display of data where necessary. However, given the amount of data requirements for the administration pages of this project, as

well as time / resource constraints, it was necessary to use features of the Data Source “Wizard” for the administration pages. Inherently, for each administrative data requirement, this added:

- Dataset (hand coded up until this point);
- Binding Source (which allows the data grid and Binding Navigator to “talk” to each other, effectively allowing both to be bound to the same data source);
- Table Adapter (to check for changes, match these against the original form of the table, and update as necessary);
- Binding Navigator (to peruse the records).

The declaration and implementation for these objects (for each data requirement) was stored in a C++/CLI header file, for example `ItemsDataSet.h`. As VS2005 uses its own XSD compiler to generate this source code, each time a change was made to `ItemsDataSet.h` and the project rebuilt, the resulting XSD compiler would “rewrite” the code without confirmation to its own interpretation and implementation – making it impossible to affect changes – specifically relative file path changes – and as such, other workarounds had to be investigated. It is of the engineers’ belief that future projects avoid automated generators of code wherever possible, although in this circumstance under the resources available it was necessary.

Controls, Objects and Processes to Enhance UI Experience

In revising the completed project, the developers noted several features that could have significantly increased the clients experience and efficiency, as well as decreasing potential error making. Although more could be implemented, a non-exhaustive list of some of the controls and objects that would increase user UI satisfaction includes:

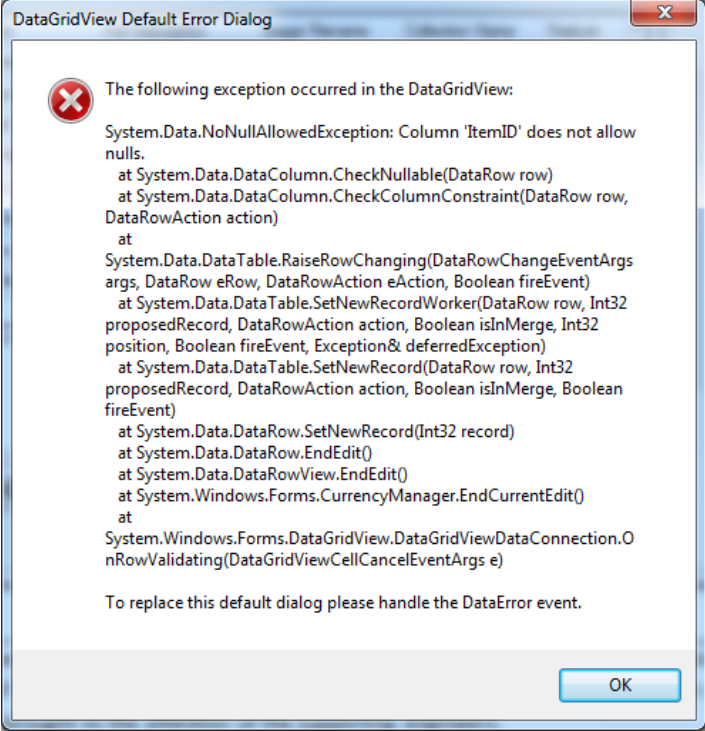
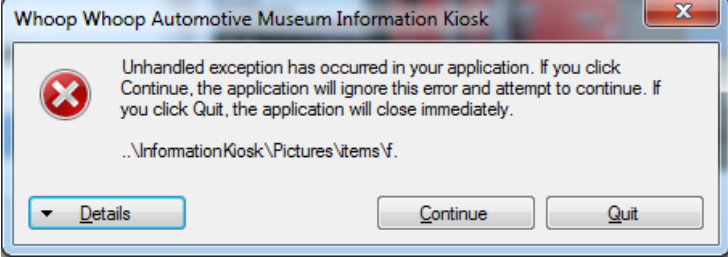
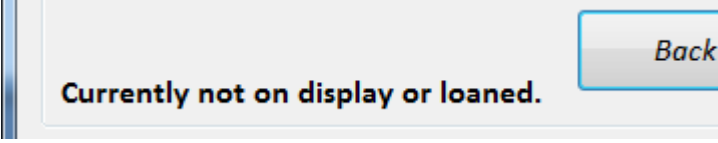
- Images to enhance the first appearance screen;
- Ability to use keystrokes - e.g. enter key when searching;
- Full help tutorial.
- Browsing features – displaying animations or images would be preferred to make features “stand out”;
- Adjustable font size and screen readers (voice over’s) to ensure equitable access;
- Hide the Administration link to a more secure area, or key code / key card access;
- Upload images (discussed above);
- Administrator Loans form - A “date picker” box for the date a loan is made;
- Screen dynamically resizes with text, as well as with monitor / image size – to “best fit” with resolution;
- Further consultation with industry contacts to better model the data – perhaps sort by year / make / etc.
- Media – audio, visual, animation, models – anything to increase the aesthetic appeal!
- Google maps style interaction for display of where loans are coming from or going to.

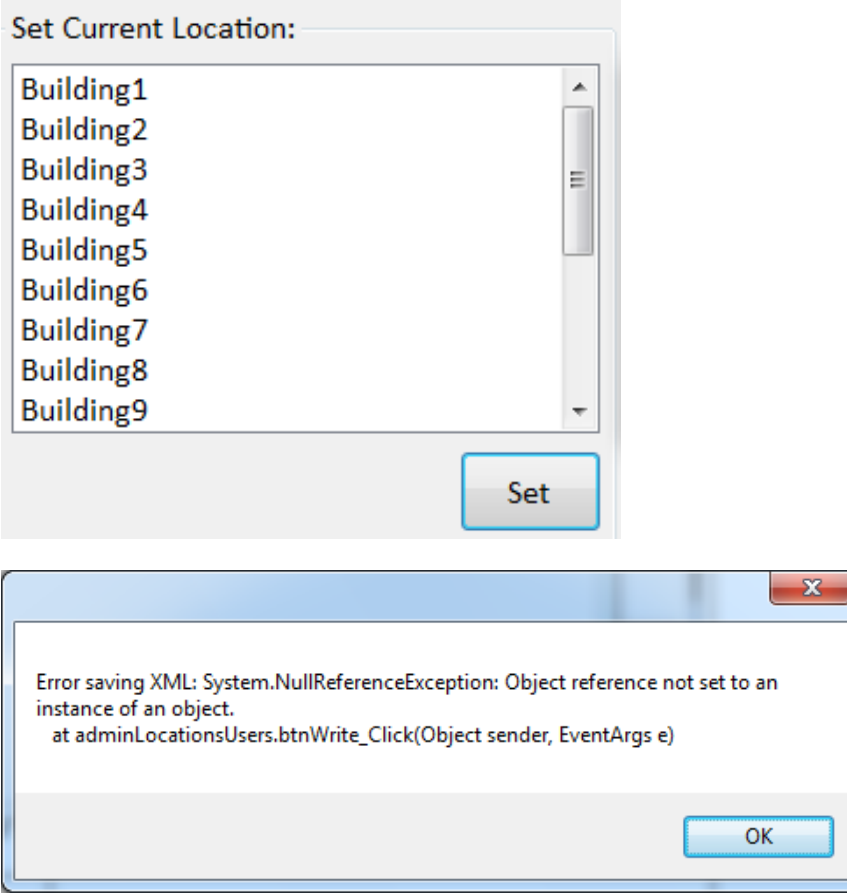
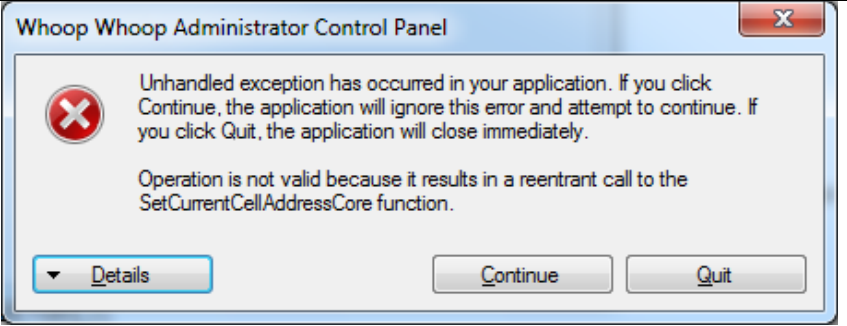
It is believed that these controls and objects would be of significant benefit to the user and administrator experience, and may be incorporated in future revisions of the Information Kiosk.

Errors at Run-time

Given the time and resource restrictions, it was the first priority of the engineers to deliver a system to the client that was chiefly concerned with reliability and performance. As such, no errors must

exist at run-time. Given the nature of the system, however low the probabilities are of the following events of actually occurring, it must be noted that the following sequence(s) of events needs evaluation and further discussion with the client.

Sequence	Error	Implication
Trying to enter a null or incorrect value for a primary key		The data grid in the administration pages will not allow the change.
Adding anything without a valid image link		Will not let a browse or more info until the link is fixed.
Adding an item without a collection		Nil – except if the item is on display.

<p>Setting a location without actually picking one first</p>		<p>The XML file will not write, as the lack of initialisation of the list object prevents the code from executing. Still, this is an exception that should have been handled – and more so prevented as has all other list box controls with initially selected items.</p>
<p>Entering the client phone form, playing with a combo box but not saving, exiting, then re-entering the phone form (again).</p>		<p>Bizarrely enough, this returns an error, although by clicking the Continue box this allows re-entry to the client phone form. It is believed that due to playing with the combo box, this row may be temporarily “locked” for editing.</p>

Given that these errors have fixes and / or workarounds, it is necessary that staff be trained in the correct use of this software, and that any unnatural (or unexpected) results not thought of here are immediately brought to the attention of the supporting engineers.

Debugging Environment Updates

During alpha testing, the system was compiled on different Visual Studio environments. The first edition of Visual Studio 2005 **would not compile** all of the required files; the building report stated:

```
1>Creating DataSet class using XSD ...
1>Writing file
'c:\Users\Engineer\Desktop\InformationKiosk\InformationKiosk\ItemsDataSet.h'.
1>Error: There was an error processing '.\ItemsDataSet.xsd'.
1> - Error generating code for DataSet 'ItemsDataSet'.
1> - Indexed property cannot have an empty parameter list:
'System.CodeDom.CodeMemberProperty'
1>Parameter name: e
```

After investigation of online forums, it was determined that the difficulty in this instance that XSD was having of compiling the *Items data set* (as opposed to all of the other data sets which were accepted by the compiler) was to do with the table named “**Item**” – the old interpreter thought this was an indexed property, not the name of a table. This bug in the Microsoft compiler was resolved with Visual Studio 2005 Service Pack 1 – which must be installed to compile this solution. Evidence of the bug and its resolution fix can be found on the Microsoft Site (2007) at this URL:

<http://connect.microsoft.com/VisualStudio/feedback/details/123535/managed-c-project-fails-to-add-a-web-reference>

Add New Buildings

The ability to add new buildings was not a requirement of the XYZZY museum; however the developers saw an opportunity to create a function where new buildings could be added. As it stands, Administration can edit and update exits from the Buildings table. However, to add new buildings the exits had to be allowed as zero-length strings (“”). To do this, a dataset was created from a copy of the buildings table with its source set to a union with the copied table and a zero length string; this result was set as a source for the exit combo boxes. Unfortunately, this deposited neither a null or empty value – the result was undefined, which in the current limitations of the WYSIWYG editor, difficult to check for. The engineers believed a solution would be to store undefined exits as the string “undefined” – unfortunately time prevented this last implementation from occurring.

Auditing XYZZY

In completing the ISO 9001 gap analysis, limited information was available on the XYZZY company procedures and protocols; including whether (or not) access was available to a company quality control manual. With limited ability to investigate this company, the review took place from the point of view of the engineers developing the Whoop Whoop Information Kiosk system; which limits the universe of discourse in which elements can be evaluated for their worth or effectiveness in a

complete quality management system. As such, if any of these manuals, procedures or documents existed without the engineers knowledge, they were not including in the ISO 9001 review.

Future Directions

Given the above discussion of requirements, work breakdown, scheduling, risk management and reflections, the engineers – Engineer A and Engineer B – gained considerable experience in programming within a .NET environment. More so, the procedures of quality assurance above, reflected in all documentation, have given both engineers a deeper understanding – and beyond this, belief – to the benefits of implementing quality processes and quality management systems. Working collaboratively in this project, the engineers demonstrated the ability to solve problems in both project management and the development of complex systems. It is hoped that the future experiences of both engineers enhance and further develop the methodology employed in this project, and (hopefully) with it will come further insights into the nature of quality management systems, client-oriented project management and protocol / standards collaboration for all facets of software engineering, including (but not limited to) planning, risk management, implementation of specified systems and future review.

References

Unless otherwise specified, this assignment was completed using only the following references:

Sommerville, I. 2007. Software Engineering 8th Edition. Addison-Wesley, Edinburgh Gate, Harlow.