Project Registry Report

Vilma Rudžionytė , Viktoras Čiumanovas ,
Darius Damalakas , Martynas Kriaučiūnas

November 8, 2005

Contents

1 Introduction to Project Report **1**

1.1 Company Analysis 1

1.1.1 Company Background 1

1.1.2 Problem 1

1.1.3 Project Constraints and Terms 2

1.2 Resources 2

1.2.1 Place and Time 2

1.2.2 Hardware 2

1.2.3 Software 2

1.2.4 Human Resources 2

1.2.5 Project Adviser 3

1.2.6 Customer Contact Person 4

1.3 Project Report Structure 4

2 Iteration 1 – Inception phase (17 08 – 22 08) **5**

2.1 Iteration Goals and Objectives 5

2.1.1 Methodology Choice 5

2.1.2 Agreeing on Development Case 7

2.1.3 Initiating Project 8

2.1.4 Artifact Templates 8

2.1.5 Risk Assessment 9

2.1.6 Risk Management Strategy 9

2.1.7 Role Assignment 10

2.1.8 Teamwork Contract 10

2.1.9 Company and Team Contract 10

2.1.10 Software Development Plan 11
2.1.11 Phase Plan 12
2.1.12 Iteration Management Plan 13
2.2 Requirements and UI Prototype 13
2.3 Development 13
2.4 Testing 14
2.5 Iteration Assessment 14
2.5.1 Are the Goals Met? 14
2.5.2 Reevaluate Risks 14
2.6 Next Iteration Planning 14

3 Iteration 2 (23 08 – 30 08) 16
3.1 Iteration Goals and Objectives 16
3.1.1 Doing Domain Model 16
3.1.2 Doing Use Case Collection 17
3.2 Development 17
3.2.1 Build Two Use Cases 17
3.2.2 Initial Database Usage Strategy 18
3.3 Testing 18
3.4 Iteration Assessment 18
3.4.1 Are the Goals Met? 18
3.4.2 Reevaluate Risks 18
3.5 Next Iteration Planning 19
3.6 Goals 19

4 Iteration 3 (31 08 – 13 09) 21
4.1 Iteration Goals and Objectives 21
4.1.1 Iteration Planning Problems 22
4.2 Requirements and UI Prototype 24
4.2.1 UI Navigation Diagram 24
4.3 Development 24
4.4 Testing 25
4.5 Iteration Assessment 25
4.5.1 Are the Goals Met? 25

5 Iteration 4 (14 09 – 28 09) 26
5.1 Iteration Goals and Objectives 26
5.1.1 Phase Plan Change 26
5.2 Development 27
5.2.1 Database Architecture 27
5.2.2 System Architecture 28
5.2.3 Test Plan 29
5.3 Testing 30
5.4 Iteration Assessment 30
5.4.1 Are the Goals Met? 30
5.5 Next Iteration Planning 30

6 Iteration 5 (29 09 – 29 10) 33
6.1 Iteration Goals and Objectives 33
6.2 Requirements and UI Prototype 33
6.3 Development 34
6.3.1 Database Subsystem 34
6.3.2 Filtering 34
6.4 Testing 35
6.5 Iteration Assessment 35
6.5.1 Are the Goals Met? 35

7 Requirements 37
7.1 Introduction to Requirements 37
7.2 First Step - Domain Model 37
7.3 One Important Mistake 39
7.4 Use Cases 39
7.4.1 Purpose 40
7.4.2 Collecting Use Cases 40
7.5 Encountered Problems 41
7.6 User Navigation 41
7.7 User Interface Requirements 42
7.8 System Prototype Requirements 42
7.9 Final Iteration Requirements 43

8 User Interface **44**
8.1 User Interface 44
8.1.1 User Interface Goals 44
8.1.2 Web Elements Functionality 45
8.1.3 Building User Interface 46
8.2 Cascading Style Sheets 48
8.3 CSS Evaluation 49
8.3.1 CSS Choice 49
8.3.2 User Interface Evaluation 49
8.4 Conclusion 51

9 Implementation **54**
9.1 Building Implementation Queue 54
9.1.1 Development Risks 54
9.1.2 Use Case Prioritization 55
9.1.3 Building Final Development Queue 55
9.2 Database Design 56
9.2.1 Problem 56
9.2.2 Possible Solutions 56
9.2.3 Our Choice 58
9.3 The Page Life Cycle 58
9.4 Dynamic vs Static 61
9.5 Building two Functional Prototypes 63
9.6 Darius Prototype – Nano Pages 63
9.6.1 Prototype Goals 63
9.6.2 Explained 64
9.6.3 Class Diagram 64
9.6.4 Advantages of this Prototype 67
9.6.5 Disadvantages of this Prototype 67
9.7 Martynas Prototype 67
9.7.1 Prototype Goals 67
9.7.2 Limitations of ASP.Net 67
9.7.3 Design 68
9.7.4 Advantages of this Prototype 69

9.7.5 Disadvantages of this Prototype 69
9.8 Prototype Choice 70
9.9 Building Arklys Injections 70
9.9.1 How to Prepare the Page to Use DymacicContent 71
9.9.2 How to Add or Remove DynamicContent 72
9.9.3 How to Create DynamicContent Web User Control 73
9.9.4 Problems Creating Arklys Injections 73
9.10 Improvements to System 75
9.10.1 New DBFillers and Improved Controllers 75
9.10.2 FilterDropDownList 76
9.10.3 FilterTextBox and DatePicker 76
9.11 Final System Design 76
9.11.1 Package View 76
9.11.2 Project File Structure 79

10 Testing 83
10.1 Introduction 83
10.2 Purpose of Testing 83
10.3 Test Types 84
10.3.1 Unit Testing 84
10.3.2 User Interface Testing 86

A Iteration 1 – 17 08 - 22 08 Inception Phase 88
A.1 Goals 88
A.2 Objectives 88
A.3 Work Orders 89
A.4 Iteration Assessment 89
A.4.1 IterationSummary 89

B Iteration 2 – 23 08 - 30 08 Elaboration Phase 91
B.1 Goals 91
B.2 Objectives 91
B.3 Iteration Assessment 91
B.3.1 Iteration Summary 91

C Iteration 3 – 31 08 - 13 09 **95**
C.1 Goals 95
C.2 Objectives 95
C.3 Iteration Assessment 95
C.3.1 IterationSummary 95

D Iteration 4 – 14 09 - 28 09 **99**
D.1 Goals 99
D.2 Objectives 99
D.3 Iteration Assessment 99
D.3.1 IterationSummary 99

E Iteration 5 – 29 09 - 28 10 Construction Phase **102**
E.1 Iteration Goals and Objectives 102
E.2 Requirements and UI Prototype 102
E.3 Development 103
E.3.1 Database Subsystem 103
E.3.2 Filtering 103
E.4 Testing 104
E.5 Iteration Assessment 105
E.5.1 Are the Goals Met? 105

F Iteration Management Plan **106**
F.1 Introduction 106
F.2 Document Holder 106
F.3 Iteration Timing 106
F.4 Iteration Plan Structure 107
F.5 Iteration Evaluation and Assessment 107

G Software Development Plan **109**
G.1 Deliverables 109
G.1.1 Supplementary Specification 110
G.1.2 Use Case Specification 110
G.1.3 User Interface Specification 110
G.1.4 Software Architecture Document 110

G.1.5 Test Plan 110
G.1.6 Initial Prototype 110
G.1.7 Functional Prototype 110
G.1.8 Operational System 111
G.1.9 Deployment Tools 111
G.2 Human Resources 111
G.2.1 Team Structure 111
G.2.2 Team Capabilities 111
G.2.3 Roles and Responsibilities 111
G.3 Project Planning 112
G.3.1 Phase Plan 112
G.3.2 Iteration Plans 112
G.3.3 Project Plan 113
G.4 Technical Process Plans and Guidelines 114
G.4.1 Iteration Management 114

H Risk Management Strategy 116
H.1 Introduction 116
H.2 Document Holder 116
H.3 What is a Risk 116
H.4 Specifying a Risk 117
H.5 Risk Management Strategy 117

I Project Management Risk List 119
I.1 Introduction 119
I.2 Document Holder 119
I.3 Risk List 119
I.3.1 Mis-conceived or Undefined Project Scope 119
I.3.2 Low-risk Tasks go First 120

J Glossary 122
J.1 Document Holder 122
J.2 Glossary 122

K Development Risk List **123**
K.1 Introduction 123
K.2 Document Holder 123
K.3 Risk List 123
K.3.1 Lack of .Net Knowledge 123
K.3.2 Inappropriate Database Design 124

L Software Architecture **125**
L.1 Introduction 125
L.2 Document Holder 125
L.3 Architectural Goals 125
L.3.1 Main Components 126
L.3.2 Database 126
L.3.3 Web Application 128

M Supplementary Specification **131**
M.1 Introduction 131
M.2 Document Holder 131
M.3 Requirements 131
M.3.1 Development Environment 131
M.3.2 Persistency 131
M.3.3 User Interface 132
M.3.4 Security 132
M.3.5 Accurateness 132
M.4 Functional Requirements 132
M.4.1 Data Versioning / History 132
M.4.2 Multi-attachment Ability 133
M.4.3 Filtering Boxes in Datagrid 133

N Use Case Specification **134**
N.1 Introduction 134
N.2 Document Holder 134
N.3 Domain Model Diagram 134
N.4 Business Logic 134

N.4.1 Task Creation Cycle 134
N.5 Use Case Template 135
N.6 UC1: Insert Project 136
N.7 UC2: Log Into System 136
N.8 UC3: Insert Task 137
N.9 UC4: View Projects 137
N.10 UC5: View Project In Detail 138
N.11 UC6: View Events 138
N.12 UC7: View Event in Detail 139
N.13 UC8: View Task in Detail 139
N.14 UC9: Edit Task 139
N.15 UC10: View Tasks 140
N.16 UC11: View My Approvals 140
N.17 UC12: Approve Task 140
N.18 UC13: Delete Task 141
N.19 UC14: Delete Project 141
N.20 UC15: Edit Project 141
N.21 UC16: Delete Event 142
N.22 UC17: Edit Event 142
N.23 UC19: Filter 143

O Company, School and Team Contract 144

P Development Case 146

Q Test Plan 155
Q.1 Introduction 155
Q.2 Document Holder 155
Q.3 Types of Tests 155
Q.3.1 Functional Tests 155
Q.3.2 User Interface Tests 156
Q.3.3 Security Testing 157

R Team Contract 158
R.1 Purpose of this Document 158

R.2 What is the Goal for the Team Work (What is to be Delivered to Who) . . . 158
R.3 Team Members . . . 159
R.4 Resources Available for the Team and Assessment of Strengths and Weaknesses . . . 159
R.5 Work Organisation . . . 159
R.6 What is the Goal for Team Work . . . 159
R.7 Technical Resources Available . . . 162
R.8 Software . . . 162
R.9 Hardware . . . 162
R.10 Working Place . . . 162
R.11 Risks . . . 162

S XP versus UP a Methodology Comparison Study Work 164
S.1 Introduction . . . 164
S.2 Philosophy . . . 164
S.2.1 Paradigm . . . 165
S.2.2 Objectives . . . 165
S.2.3 Domain . . . 165
S.2.4 Target . . . 166
S.3 Model . . . 166
S.4 Techniques and Tools . . . 166
S.4.1 UP Techniques . . . 166
S.4.2 UP Tools . . . 167
S.4.3 XP Techniques . . . 167
S.4.4 XP Tools . . . 167
S.5 Scope . . . 168
S.6 Outputs . . . 169
S.6.1 Introduction . . . 169
S.6.2 UP Outputs . . . 170
S.7 Practice . . . 171
S.7.1 Background . . . 171
S.7.2 User Base . . . 171
S.7.3 Participants . . . 171

S.7.4 UP Participants 171
S.7.5 XP Participants 171
S.8 Product 172
S.8.1 UP Products 172
S.8.2 XP Products 172
S.9 Conclusion 172
S.10 References 174

T Project Registry Source 176
T.1 Pages 176
T.1.1 BatchView Pages 179
T.1.2 Insert Pages 219
T.1.3 Edit Pages 251
T.1.4 DetailView Pages 259
T.1.5 Administrator Pages 265
T.2 Users Management 267
T.3 Common Library 317
T.4 Custom Controls 324
T.5 Tests 332

Chapter 1

Introduction to Project Report

1.1 Company Analysis

1.1.1 Company Background

"Blue Bridge" (later BB) is a big IT company, which possess big part of Lithuanian Information Technology market. The company is doing projects from various areas of IT.

"Project Registry" is an information system, which purpose is to manage companies human resources among many projects.

1.1.2 Problem

Our task in BB is to develop a new version of "Project Register" system. The system collects information about all projects, their tasks, assigned hours for execution, employees, and project/department managers in the company. "Project Register" can also provide statistical information, which helps manage the company's resource usage.

1.1.3 Project Constraints and Terms

The project will be carried out in agreement with the following constraints and terms:

Time: 2.5 months (from 17 of August to 31 of October),

Human resources: 4 team members.

System: Must be developed using ASP.NET and C#.

Main requirement: Usability.

1.2 Resources

1.2.1 Place and Time

The team was given a small room in companies office and was allowed to work any time from Mondays to Fridays in the office.

1.2.2 Hardware

BB provided the team with two stationary PCs with internet connection in the office. The other two team members had their own laptops. All computers were connected to intranet.

1.2.3 Software

The company also provided us with:

Software: MSDN Library, Visual Studio.NET Enterprise Architect 2003, SQL Server 2000 and MS Project;

Old System: BB made a copy of an old system and gave us access to it through web interface with administrator privileges.

1.2.4 Human Resources

The team consists of four team members (see table 1.1).

We are probably the best team and here are our weaknesses and strengths:

Name	Telephone	E-mail
Vilma Rudžionytė	+37060068449	vilma.rudzionyte@gmail.com
Martynas Kriaučiūnas	+37067002732	martynask@gmail.com
Viktoras Čiumanovas	+37067485615	viktoras.ciumanovas@gmail.com
Darius Damalakas	+37060246428	darius.damalakas@gmail.com

Table 1.1: Team Members

Name	Good Traits	Bad Traits
Vilma Rudžionytė	Raises teams morale 200%	Does not program, changes often study programs
Martynas Kriaučiūnas	Very structural thinking, Good at addressing technical problems, overcomes obstacles alone	Takes too much jobs at a time
Viktoras Čiumanovas	Keen on broadening his knowledge, Talkative, Open-hearted	Spends too much time on contemplation on his tasks
Darius Damalakas	Easily reacts to any kind of changes, Finds solutions alone,	Can't work steady for a long time, not always able to accept other people with their good and bad sides, under-estimates own limits and abilities

Table 1.2: Team Capabilities

1.2.5 Project Adviser

Our project adviser is Michael Claudius:

Name	e-mail	Phone number
Michael Claudius	claudius@rhs.dk	+4588523331

As we are doing our project outside Denmark, we are only able to communicate via e-mail or internet telephony. This was an obstacle to communicate

often and productively. However, no other better alternatives were found.

1.2.6 Customer Contact Person

Name	Telephone	E-mail
Andrius Dienys	+370 699 97017	Andrius.Dienys@bluebridge.lt
Dalius Mašalas	+370 685 41445	Dalius.Masalas@bluebridge.lt

1.3 Project Report Structure

Our project report is divided into chapters. The first five chapters discuss the project in perspective of time. Each of the first chapter is dedicated for only one iteration.

Subsequent chapters are related to one or two similar practices, showing in-detail what was happening in the project in particular area.ening in the project in particular area.

Requirements: This chapter discusses what decisions were made and thoughts contemplated about collecting and eliciting requirements.

User Interface Prototype and Evaluation: User Interface related activities encompass prototype building and evaluation, as well as communication with customer.

Implementation: Chapter focuses on discussing what problems occurred whilst implementing the system and what solutions were applied. The chapter is concerned only about technical aspects of implementation.

Testing: This chapter focuses on discussing what problems occurred during the code preparing for testing and they were solved.

Chapter 2

Iteration 1 – Inception phase (17 08 – 22 08)

2.1 Iteration Goals and Objectives

Inception phase in our project was dedicated for getting used to new environment, get acquainted with company, install and find any necessary software and agree on development case.

The goals were to:

1. Agree On Methodology
2. Agree On Development Case
3. Initiate Project

2.1.1 Methodology Choice

One of the most important things in the project – Methodology choice. So in this section we will discuss:

- Goals of methodology
- Requirements for a methodology

Methodologies Goals

"In software engineering and project management, a methodology is a codified set of practices (sometimes accompanied by training materials, formal educational programs, worksheets, and diagramming tools) that may be repeatably carried out to produce software. Software engineering methodologies span many disciplines, including project management, analysis, specification, design, coding, testing, and quality assurance. All of the methodologies guiding this field are collations of all of these disciplines."[1]

In addition, a methodology should specify the training needs of the user of the methodology. So, why we need methodology? Here we can identify three categories of rationale:

1. A better end product – naturally we wanted to have a good system (product). For this reason we had to compare the results of using different methodologies.

2. A better development process – productivity is enhanced with the use of a methodology; that is, we can either build system faster, given specific resources or use fewer resources to achieve the same result.

3. A standardized process – needs of having a common approach throughout an organization; More integrated system can result (easily change from project to project without retraining).

So, the main goal of the methodology is to help control projects writing.

Requirements for the Methodology

We have chosen methodology according to requirements from the company and our group.

- Company – from the beginning was important requirement: to use an iterative process.

- Our team – The team insisted on going deeper into RUP, opposing that the previous semester project was following XBreed, a well integrated solution between Scrum and XP.

[1] http://en.wikipedia.org/wiki/Methodology_(project_management)

In the 3rd semester we have studied comparison of methodology issues. There are a lot of methodologies and also there are criterias that might be considered in assessing methodologies: rules, simplicity, validity of design, documentation standards, etc.

XP and UP are well-known methodologies. Furthermore, they are very different. The comparison of these two methodologies is in Appendix S

Conclusion

After compare methodologies, we decided to use UP methodology since:

- We have to use iterations
- We need take more care of the project

2.1.2 Agreeing on Development Case

Establishing Development Case

Development case describes the software-development process for a project. As defined in RUP, the development case captures the tailored process for the individual project.

Project manager, in communication with the team, made a development case in a form of simple presentation document. The goal was to agree on what artifacts and which activities are essential throughout the project.

The development case is included in appendix P

Development Case Lessons

"Road to hell is full of good intentions" – a saying which fits perfectly our situation.

The intentions of development case were good, but whomever did this development case, lacked a big deal of competency and experience to do so at least satisfactorily.

The goal of the document was from the beginning unfeasible. That is plainly visible (now, after we experienced it on our own skin), by simply looking and estimating our project. RUP, essentially, is not suited for small projects, the number of people should be no less than 10, or the time of the project should be

as long so as to the project could actually run the intended processes. Now, in project like ours the number of artifacts should be manageable and maintainable. However, we chose to do approximately two or three documents for each person. What is more, these documents were considered to be used "internally" in the team, which ultimately proves that this was not ever going to be working.

So the lesson is that it is essential to walk on the pathway which goes on the mountain peak and not to fall down. The pathway is the most harmonious choice from the following sayings:

- Do the work which actually does the job and not the ones which makes you go circles.
- Control the process and make it predictable.

In our case, we did the many artifacts, and not the things, which actually bring value to the customer.

2.1.3 Initiating Project

To initiate the project we had to agree on artifacts to be produced. We took the list of documents from our development case (P). We had to prepare template for the following items: Risk List, Risk Management Plan, Software Development Plan, Use Case Model, Supplementary Specification, Team Contract, Test Plan.

Risk assessment, role assignment, risk management and iteration management plans are all integral activities in this phase.

Teamwork Contract is a means of transferring each members responsibility to a formal level.

For Project Management had to develop iteration and phase plans.

2.1.4 Artifact Templates

Every document must have an introduction and document holder clause. The first one describes what is the main purpose of the document. This was necessary, as the number of managed document was high. The second states who is responsible for managing that document. Very good for blaming somebody for anything.

2.1.5 Risk Assessment

In the establishment of the project phase the most important thing is to establish a project in a way, that it allocates resources in the best way and assigns responsibilities for correct people.

Having incompetent people doing irrelevant job, soon will conclude in a project going havoc. And people without resources will simply soon be performing a play, with the audience either acting together, or resisting and blaming of doing inappropriate and inadequate job.

Thus in the first place what is necessary to do is to:

1. Define roles to ensure that proper responsibility area is given for each team member (as much as possible)
2. Sign group work contract to establish strict (but broad, for the members could play well) rules and define limits and constraints.
3. Assign authority and resources to make process possible to be flexible and organized.
4. Define The process evaluation procedure.

2.1.6 Risk Management Strategy

Risk management strategy was defined in a document "Risk Management Strategy" (H).

The goal of the document was to agree in team on how all the risks will be managed during the project.

This document emphasized, that we agreed to separate project managements risks aside from development risks and that these will be held and deal separately.

The document also explained how an identified risk must be documented. It noted, that to clearly identify the risk, the member must write the name and description of the risk, severity and probability, risk status, risk manager and the mitigation strategy to be used.

2.1.7 Role Assignment

Role assignment, ideally, should reflect the inner values of the team member. This belief is based on our personally experienced facts that human beings are more than machines and we are in need of not only energy of matter (such as food), but energy of many forms, including sleep, and, more important, anything else, which could be grouped to term "psycho-energy". This term encompasses everything, which we can not observe scientifically, but of which we can be assured by simply examining our own experiences.

As for example lets consider a small team (say, 4 to 5 members), and two situations. One is where team members are in light-weight atmosphere, full of good mood, and full of freedom, where team members feel free to express their identity in many ways. Another situation is where team members are closed, for they do not feel free to express theirselves in ways they feel is right to do so.

The situations are very in contrast, and the reader can easily ascertain what will be the results if had any similar experience. Will the team overcome easily obstacles, and how well any process will be carried out is naturally dependent not only on resources and assigned roles, but how do they fit their personalities.

The roles in our team were not defined artificially, i.e. the team naturally observed and noted that one member is more prone to prompt action, the others are more prone to anger, and the third one is natural idler.

The table Team Roles 2.1 shows member role assignment.

2.1.8 Teamwork Contract

In the beginning we have created Team Contract. It helped assigned and define team roles and makes all members more responsible. All Team Contract is placed in Appendix R

2.1.9 Company and Team Contract

One of the important things was also to make contract between company, our team and school. This creates guarantee for the company that we do not leave the project before the end of the project. Also it guarantees that the project will be done completely. The contract is placed in Appendix O.

Name	**Main Role**
Vilma Rudžionytė	User Interface Evaluation, Project Report Designs and evaluates UI
Description	She has done UI design in 3rd semester. She easily copes with small-scope tasks, but easily bumps into dead-ends when given more abstract tasks. Needs strong guidance
Martynas Kriaučiūnas	**Software Architect:** Responsible for setting and assessing architectural analysis, defining reuse strategy and leading Design and Implementation Work flow
Description	He has outstanding experience in designing software and building reusable architectures. 4 years of experience with real world projects. Critical thinking (some times too much). Structural thinking
Viktoras Čiumanovas	**User Interface Design:** Responsible for building UI prototype and communicating with customers
Description	He has experience in designing software and building reusable architectures. Critical thinking. He has experience in interaction with customers.
Darius Damalakas	Project Manager, Process Engineer Responsible for guiding the project and ensuring everybody agrees on the working methods
Description	He has outstanding experience in designing software and building reusable architectures. 4 years of experience with real world projects. Critical thinking (some times too much). Structural thinking

Table 2.1: Team Roles

2.1.10 Software Development Plan

For captured our roles and responsibilities, phase plan and iteration plans, a software development plan. This is one of the key artifacts, as defined by our development case. SDP is the starting point for anybody to search for informa-

tion about the project.

Phase Planning

In the beginning we optimistically planned to divide project into two phases, or even two related projects. One part would focus on porting the system to .Net architecture, and this we were considering to finish till 21st October. The last week we left out unplanned.

You can see our phase plan in table 2.2

2.1.11 Phase Plan

The project is going to be developed through 17 08 2005 to 31 10 2005. This time was split into four phases.

Date	Phase Name	Phase goals
18 08 2005 – 22 08 2005	Inception	Establish project and agree on development case
23 08 2005 – 28 09 2005	Elaboration	collect requirements, define user interface, define architectural design, implement an architectural prototype
29 09 2005 – 21 10 2005	Construction I	Build an operation system, test system
21 10 2005 – 31 10 2005	to be planned later	...
31 10 2005 – 07 11 2005	to be planned later	...

Table 2.2: Phase Plan

Project Planning

Immediately, we were thinking of making iterations very short, so we could gain more control over them, by simply doing more evaluation and more planning.

Thus, we were planning to do more than six iterations. One for inception phase, three for elaboration and two or three for construction.

2.1.12 Iteration Management Plan

Iteration management plan solved problems of how to manage each iteration, how to write iteration plans and how to do iteration evaluations.

The plan stressed that iteration planning comes on Tuesdays participating all team members, and that meetings with customer occur on Wednesdays.

Plan iteration management also defined iteration plan structure. Each iteration must define its goals and objectives, where goals can be understood as high-level objectives.

What is most important, this document described how the process of iteration evaluation must go. It emphasized, that during iteration evaluation not only the goals and objectives must be achieved, but also that iteration must be documented and summarized. This would help us collect information about our project and use it for learning.

2.2 Requirements and UI Prototype

In this activity, all team members intensively were using the system to get used to it and know what is going on and how it works. Aside from this, no other important decision were made or performed.

2.3 Development

As what concerns the development area, the team was preparing working places by installing various software. There also arise some problems with ASP.NET installation on laptops. It was solved further iterations. There were some problems with communication among computers: they were invisible from each other. This problem was also solved later.

2.4 Testing

Not performed in this iteration

2.5 Iteration Assessment

In iteration end, we had an iteration assessment. The iteration assessment was conducted as was defined in our Iteration Management Plan (F.5).

For the 1st iteration, the iteration evaluation is included in appendix A.4.

2.5.1 Are the Goals Met?

Our goals were three-fold: to Agree On Methodology, Agree On Development Case and Initiate Project.

Methodology was discussed in section 2.1.1. Later, the methodology was tailored to our specific project by building a development case, section P.

Project initiation was a multi-task, composed of many activities, such preparing contracts both between team members (Team Contract 2.1.8), and a three-sided contract between team, company, and the school (2.1.9). We also did requirement management strategy, risk assessment, created risk management strategy, assigned roles and made project planning, which resulted in both the phase plan, and the next iteration plan (2.6).

2.5.2 Reevaluate Risks

All risks have been identified in this iteration.

2.6 Next Iteration Planning

Next, second iteration was considered to go to the next phase – elaboration. It would last from 23rd of august to 30th of august.

Initially, we planned this iteration to achieve these goals:

1. Establish a good base for requirements
2. Build an initial implementation prototype

More precisely, we intended to collect use cases, build domain model and try to program first use case (UC1) Create New Project and UC3 Edit Project web pages, of course, with only basic functionality. A requirement for implementation was that the system must already collect data from a database.

The iteration plan is included in appendix B.

Chapter 3

Iteration 2 (23 08 – 30 08)

3.1 Iteration Goals and Objectives

The main goals of iteration 2 were to:

- Collect more use cases,
- Build domain model,
- Implement two use cases,
- Develop Database usage strategy.

3.1.1 Doing Domain Model

The initial domain model captured the most important aspects of the domain of problem. It is shown in figure 3.1. We did it mainly looking at the demo system we had, and transferred the most important concepts to diagram.

There were a lot of disputes about the notation of diagram, but finally we agreed, that domain model will show:

- Real world classes with their attributes and attribute types,
- Multiplicity between class relationships,
- Any inheritance relationships exist.

Such a domain model established a correct understanding of the system between members.

Figure 3.1: Domain Model

3.1.2 Doing Use Case Collection

Use case collection was performed during meetings of team and customers. The collected information (which was written down during meetings), afterwords was added to iteration evaluation phase. Here is a sample of what we collected. The full list can bee found in Iteration 2 Evaluation section in appendix B.3.1.

3.2 Development

3.2.1 Build Two Use Cases

The goals in the development were to build first use case *UC1: Create New Project* and *UC3: Edit Project* use cases. The requirements was that the system would already connect to database and show data from it.

Doing this task we faced with a severe problem of lack of .Net knowledge. The team had to study intensively .Net platform features and C# language.

Soon, before iteration end we succeeded in connecting to database, making

the two use cases with very small functionality. We also found out how to do sorting of data in datagrid, how to do edit pages, and how to insert many control in one single data grid cell.

3.2.2 Initial Database Usage Strategy

Another goal was to develop a database usage strategy. This database usage strategy was discussed with Dalius Mašalas a few times and a good initial draft was established already in this iteration. It was finished in next iteration, and you can see it in appendix L.3.2.

3.3 Testing

Not performed in this iteration.

3.4 Iteration Assessment

In this iteration we have discussed about use case collection and building. As was mentioned in iteration gaols, we have done Domain model and discussed it.

3.4.1 Are the Goals Met?

The objectives were to create domain model (N.3). Domain model enabled us to understand the problem domain equally between all members.

Initial Database Schema was made and it was communicated with Dalius Mašalas for approval. We prioritized our development tasks (9.1.3).

Initial software architecture document was created. Prototype is now connecting to database and shows data from it.

3.4.2 Reevaluate Risks

There are many risks in every project, and thus one must evaluate carefully them. Performing any actions the first thing must always keep in his mind and be conscious about it is not to over do things and exaggerate them and not to omit them too much.

Being conscious about it, the most severe and important to document and note risks that we consider are these:

- Mis-conceived or undefined project scope

  The team has agreed to port the existing system to .NET platform until 1st of October. The problems is how to evaluate, when the project is successfully ported. What functionality must be present and which not? Which functions are considered as "add–on" functions and will be tackled in Construction phase II (see project phase plan G.3.1). What about bug penetration? Must the port be completely bug-free? Is this feasible at all?

3.5 Next Iteration Planning

According to our iteration management plan, we planned our next iteration as we did earlier: iteration is one week length and do the most riskiest tasks first. This way, all team concluded on the following next iteration goals:

3.6 Goals

1. To generate an architecture, and build a proof of concept, if feasible
2. To look at alternate solutions (Ajax)
3. To master C#

However, this plan was evaluated not as we expected by our customers. They suggested to concentrate more on system requirements collecting and old system understanding.

Summary of meeting with the company

- Do we have to implement Administration feature in the system during first construction phase? *YES.*
- Do we have to implement various reports (Ataskaitos) in the system during the first construction phase? *NO, not necessarily.*

- Do we have to implement multi-file attachment feature during the first construction phase? *NO, not necessarily.*
- In Administration− >Personal Workers Information is "Work" column. Who manages this column: Administrator or the system counts it dynamically ? *Administrator. It is important value, by which system knows to show worker in employee lists or not.*

Chapter 4

Iteration 3 (31 08 – 13 09)

4.1 Iteration Goals and Objectives

Our initial iteration goals were very focused on learning C# and how to implement one or another feature. However, this was very unacceptable by our customers.

They criticized us for focusing on not the right tasks. They argued this by pin-pointing our "progress" in customer requirement collection. By the start of September, we had done very little to collect and understand user requirements. We had conducted interviews with customer, but the Use Case Specification document was not reviewed by customers at all.

Therefore, we changed our iteration plan, and developed a new one. Now we have an old, obsolete iteration plan, and a new one (appendix C).

The new revised iteration plan included more focused activities on collecting user requirements:

- Collect user requirements,
- Develop system prototype.

The tasks in iteration 3 were:

- Complete a **User Interface Specification**: for each of 19 pages build a window with visual studio and insert the screen shot of that page into UI specification.

This, after discussion in team, was agreed to be irrelevant for our project. More important would be to do User Interface Prototype with some rapid development tools.

- Build a SAD. This includes generating DB schema, producing class diagrams, system sequence diagrams, package diagram and writing programming guidelines (such as how to use DB).
- Build DB scripts. DB scripts should create all tables, add all constraints and all relationships into database.
- Build Use Case Specifications. This includes defining and detailing all currently known 17 use cases. Use cases must be of deliverable quality to the customer. Any important vocabulary (such as who is employee and who is project manager) must be defined in glossary.
- Build **Core package**, **Permission**, **ProjectBuilder**. This includes building a NanoBuilder, NanoPageLoader, BBPage classes, CBProjectDetail and NanoProjectBuilder and NanoMyProjectBuilder 9.6.

4.1.1 Iteration Planning Problems

Mis-conceived Scope

We had a problem with iteration planning of mis-conceiving the nature of iteration. During project establishment, we discussed the question of how long should the iteration be? There were two opinions: one was to make iterations length for one week, for the sake of increased flexibility. The other was to make it longer, and at least two weeks length, putting as an argument a notion that the act of planning an iteration would take a lot of time if we had only a week for each iteration.

The group decision was to do one week iterations, which later proved itself to our project manager, that one week is simply insufficiently low. Having one week iterations is more to a control check-points than to an iteration. The team had a discussion and project manager politely excused for his dismal failure to control the project and decided to have two weeks length iterations from now on.

Poor Task Assignment

The poor task assignment problem was very intense during the beginning of project, since the project manager tried to plan and assign all tasks fine-grained. This was partly due to mis-conceivement of the nature of an iteration. As an example see table 4.1 which shows objectives of an old iteration 3.

The objectives are very unspecific and unclear, vague. More to it, they do not result in any tangible artifact, only part of in best case.

This suggests two things: either the iteration must be longer and objectives must result in one or another product, or this is simply a task list, which is for our small project way too much fain-grained.

Objective	**Worker**
Fill SAD architectural goals section	Martynas Kriaučiūnas
Evaluate Risk: less time for programming	Darius Damalakas , Vilma Rudžionytė
Generate a framework idea and build a proof of concept solution	Martynas Kriaučiūnas , Viktoras Čiumanovas
Look at system permission models and revise DB schema	Martynas Kriaučiūnas

Table 4.1: Obsolete Iteration 3 Objectives

Lessons Learned

The lessons learned are outlined in the following list:

- An iteration is more than a check-point, and it is less than a milestone. If check points are for checking are we on the right track, milestones strictly define our goals and time limits, then iterations are just right in the middle between these two.
- Iterations should produce tangible defined products. If they can be measured, even the better.
- The iteration length varies greatly from project to project. To do it right, one must evaluate (based on his/her experience) the size of the project

and the number of people working on it, and note that iteration is not a milestone nor a check-point, and it must produce tangible products.

- The plan for the iteration is the one that is specifically made for it. It should not be too detailed, nor it should be too abstract. We have check-points and milestones for such plans.

4.2 Requirements and UI Prototype

4.2.1 UI Navigation Diagram

In order to agree on how many pages and what are they and what are navigational directions, we built an User Interface Navigation diagram. The diagram is shown in figure 4.1.

Figure 4.1: UI Navigation

4.3 Development

In iteration 3 development was very intense. Darius built his own "Nano Pages" architectural prototype, and later on (pressed by Darius) Martynas built his. A prototype should be understood in this case as a one or two days implementation work, which outlines the main ideas of how the system should be built and how should it operate.

To build two different prototypes was a necessity. The main factor was that Martynas was not satisfied with Darius's solution. This dissatisfaction, as we can now observe, arouse from il-structured solution. However, on the other hand, Martynas, by that time could tell no way how to do it good.

Eventually, Martynas's prototype was built and chosen, as his solution had an initial structure. Later, it was enhanced with features, that were lacked compared to Darius solution. This is discussed in detail in section 9.8;

4.4 Testing

Not done in this iteration.

4.5 Iteration Assessment

4.5.1 Are the Goals Met?

Iteration 3 was concerned about building two user prototypes. These were done and are discussed in 9.6 and in 9.7.

Summary of meeting with the company

- Is it preferable, that for every attached file would be possible to specify a file type? - YES. It must be a list of structured titles, such "Agreement", "Analysis Result", etc.
- Do we need to create individual table with "My Events" information as is done with "My Projects" and "My Tasks"? - No, it is not needed, since there are not so many events. Who can perform all actions with events? - Project Manager, Department Manager and Employee who creates an event.
- *Suggestion: During the project: 2005 09 07, User Interface prototype review*
- Do we have to implement the Back button in the system, since it is implemented in browser too? - It is not necessary. If you be able to ensure good system functionality without Back button you may skip it.

Chapter 5

Iteration 4 (14 09 – 28 09)

5.1 Iteration Goals and Objectives

1. To create best Use case solution and finish with them,
2. Establish the following documents:
   - Database architecture,
   - System architecture,
   - Test Plan,
   - Use Case.
3. Transfer partial User Interface on the system (e.g. shift, align, etc.) and to run system prototype for the clients and other users.

5.1.1 Phase Plan Change

Analysing our progress and looking ahead we encountered our phase plan to be out-of-sync with the actual situation.

The project was not going as expected, and our belief we would finish the system before the end of project end time did not prove out.

We chose the simplest solution above all. As we are limited by time, and constrained by our knowledges, we lowered our commitments to customers and re-estimated our capability. The new phase plan was developed as shown in table 5.2

To make Database architecture documents	Martynas Kriaučiūnas
System architecture	Martynas Kriaučiūnas
Make Test plan	Viktoras Čiumanovas and Martynas Kriaučiūnas
Finish Use case and make documentation	Darius Damalakas , Vilma Rudžionytė
Continue System core programing	Martynas Kriaučiūnas and Darius Damalakas
Make User interface	Viktoras Čiumanovas and Vilma Rudžionytė
Make Initial prototype for testing	Martynas Kriaučiūnas

Table 5.1: Iteration 4 Objectives

Date	**Phase Name**	**Phase goals**
18 08 2005 – 22 08 2005	Inception	Establish project and agree on development case
23 08 2005 – 28 09 2005	Elaboration	collect requirements, define user interface, define architectural design, implement an architectural prototype
29 09 2005 – 31 10 2005	Construction	Build an operation system, test system

Table 5.2: Phase Plan

5.2 Development

5.2.1 Database Architecture

Because there is a lot of complex (e.g. maintaining relations between data, filtering it etc.) work done with various data in the system, it was decided to use a Database for storing persistent data.

Usually database architecture is done by adding relation and some extra fields to a system domain model. In our case it was a bit more difficult because there were a requirement to store history. For storing history three possible

strategies has been proposed for the client.

Log a Field Change. Using this strategy a change of a field is logged storing its old and new values in an external database table.

Duplicating Entries. Using this strategy if a table row is changed it is duplicated regardless one or more fields are altered.

Split Tables. This strategy is similar to previous one. The difference is that all tables are divided into two. The first one holds data which does not change or changes are not logged. In the second table data which has to be logged is stored. This table is managed the same like in the second strategy.

Advantages and disadvantages of the strategies are evaluated in chapter 9.2.

After analysing possible strategies it was decided to take the second one. It was considered the best, because it brings biggest advantages, least disadvantages, is easy to understand and implement.

5.2.2 System Architecture

The initial architecture for the system was made during this iteration.

In last few iterations we were learning intensively *ASP.NET* and trying to make something working (e.g. connect to a database, insert record, etc.).

Darius and Martynas had two different opinions how to design the system. Since it was difficult to compare advantages and disadvantages of both variants it was decided to implement few user pages using using both approaches (Darius and Martynas did it) and take better looking one.

Darius' System Design – Nano Pages

The system was divided into the following components: BBPageLoader, CodeBehind (CB), NanoBuilder and PermissionManager.

BBPageLoader acts as mediator, who controls the building process of web pages.

CodeBehind component classes together with .aspx files constitutes the web pages. NanoPageLoader intercepts the load routine of web pages, and then dictates the building process.

NanoBuilder, on request of BBPageLoader, "builds" the web pages. "Building" actually means modifying and configuring web pages, and not building those from scratch. This is done this way because the goal is to mix the dynamic building and visual design.

PermissionManager simply acts as a service provider. All it cares about is user identity and permissions.

Martynas' System Design

When making prototype, Martynas was concerning on code reusing and MVC (i.e. Model, View, Control) architecture. He considered these requirements to be most important:

- User interface must be designed visually (at least major part of it).
- Reduce duplication of code. It is very important, because *copy & paste* is the source of most bugs since you often forget to fix some places when refactoring.

The system was divided into the following components: DataBaseControllers (did the work concerning data loading, saving, versioning. . .) – logics, and Pages – view part. View part were structured further into: BatchView, Edit, View.

Conclusion

We chose to stick to Martynas prototype because of his good structuring of files. Darius prototype maybe had system separation into different modules, but that was not reflected in the prototype on how the system was divided into physical files. This would have had a significant impact of how the team would work, thus Martynas prototype was considered as a better one.

Read more about each prototype in chapter 9.5.

5.2.3 Test Plan

The test plan contains information about the purpose and goals of testing within the project. Additionally, the test plan identifies the strategies to be used to implement and execute testing and resources needed.

The test plan was made and such areas for testing were distinguished:

- functional – ensure proper functionality, including navigation, data entry, processing, and retrieval.
- user interface – ensure that the User Interface provides the user with the appropriate access and navigation through the functions and all the pages have a common style.
- security – ensure that only those users granted access to the system are capable of accessing the system.

5.3 Testing

Not in this iteration

5.4 Iteration Assessment

5.4.1 Are the Goals Met?

User interface prototype already contains 5 of 20 pages, which are agreed and approved by the customer.

Software architecture document L now clearly defines the architectural goals and the overall system design. Database architecture was negotiated with Dalius Mašalas and written down.

5.5 Next Iteration Planning

In our final iteration we were able to roughly estimate and calculate how much work we will be able to do till the end of our project.

We counted and estimated, that to project finish we will be able to do the following:

- Build 5 Batch View pages. Each page would take one day of programming.
- Build 5 Insert–Edit pages. Each page taking 0.7 days to program.
- Build 5 View Details page, for each page taking 1 day.

Counting all pages, we found that it would 17 days to do all these tasks. We are two programmers in the group, and work only half a day for the project, so, starting from 5th of October, we would finish on 28th of October.

However, this would happen if all would go as planned. If taken into account that:

- We need to do testing as well,
- Write Project Report,
- And some pages would take more time to implement than expected due to their complexity (for ex. some pages will require Java Script programming),

we do not have enough time to finish everything on time. Therefore, we lowered the number of pages and committed to build 4 Batch View pages (Batch Employee View, Batch Event View, Batch Project View and BatchTask view), build 4 insert–edit pages (Insert Employee, Insert Event, Insert Project, Insert Task) and build only one View page: View Project. This decision was made considering most difficult to implement pages do first and easiest last.

Our final commitment to customers looks like this: From 5th of October to 29th of October build 4 BatchView pages, 4 Insert–Edit, and 1 View Page.

Final assessment meeting will be held on November 2nd (Wednesday).

Summary of meeting with the company

- *Suggestion during the project :2005 09 14, User Interface prototype review*
- It is preferable to implement the Task insertion form like line (something like in Excel).
- The User must have possibility to choose how many Task insertion fields he/she would like to see (i.e. 10, 15, 20 etc.).
- *Suggestion: During the project: 2005 09 21, User Interface prototype review*
- The Task column in Projects table can be some images, since it would save some place and would be more understandable. It is better to implement

the column Project finished with words (Yes/No) or some images, instead with check boxes.

- Team members must add two new columns into Projects' table: Defined project hours and used project hours. Those columns must be placed after End date columns.

Chapter 6

Iteration 5 (29 09 – 29 10)

In iteration 5 the planned was remade for the last time and we committed to do the job till the end of our project.

6.1 Iteration Goals and Objectives

1. Implement system,
2. Improve User Interface prototype,
3. Write project report.

Implement Task Insert page	Darius Damalakas
Implement filtering	Martynas Kriaučiūnas and Darius Damalakas
Make User Interface prototype	Viktoras Čiumanovas and Vilma Rudžionytė

Table 6.1: Iteration 5 Objectives

6.2 Requirements and UI Prototype

According to our customer, User Interface is most important part of this project. So at this iteration a lot of work was done making and tuning user interface.

The highest priority was given to Insert Task page, because it was used very often and was inconvenient. The requirement was to make task inserting as fast

as possible. So the page was made similar to a *Microsoft Excel Spread Sheet* and it was made that few tasks could be inserted at once.

In this iteration was agreed on a common design of *Batch View* and *Insert/Edit/View* pages.

All the designing was made using CSS (*Cascading Style Sheet*) technology, so it was easily applied on the real system.

More about User Interface design is written in chapter 8.1.

6.3 Development

In the last iteration system design was made and the development was started using it. During the development process we realised that system architecture was not good enough and it was changed.

6.3.1 Database Subsystem

First of all *Database Controllers* were split into *Database Fillers* and *Database Controllers*. The task for *Database Fillers* was only to fill a table with data. The need of them came up when we realized that few *Database Controllers (old ones)* need to fill in the same data (e.g. most pages required data about projects), so we decided to reduce duplication of code by extracting parts of it into *Database Fillers*. *Database Controllers* made was responsible only for inserting new records and updating existing ones.

6.3.2 Filtering

When designing User Interface prototype, there was a request to add filtering fields above each column and it was required that filters would work while navigating through pages.

Filters influenced system architecture as well, because it system had to load filters on page load and gather data according to them.

It was decided to store filtering rules in browsing *Session*. To implement data filtering successfully, we had to create our own custom web controls:

- *FilterDropDownList* – This control was an extension to a normal drop

down list, except it had a few new attributes: FilterOperation, FilterColumnName, DefaultValueID and DefaultValueText.

- *FilterTextBox* – This control was an extension to a normal text box, except it had a few new attributes: FilterOperation, FilterColumnName.

6.4 Testing

In this iteration testing was started.

Our testing process was done according to guidelines stated in document "Test Plan" Q. It was planned to have such tests:

- functional – ensure proper functionality, including navigation, data entry, processing, and retrieval.
- user interface – ensure that the User Interface provides the user with the appropriate access and navigation through the functions and all the pages have a common style.
- security – ensure that only those users granted access to the system are capable of accessing the system.

Security testing has not been done, because only few security features has been implemented till the end of the project.

In the beginning constant refactoring often resulted in non functional system which bothered other team members. After implementing unit tests, a developer could easily check if everything worked correctly before committing changes to the main repository.

We benefited a lot from testing, because bugs were caught faster and we were always sure that the system is always well functioning.

More about testing is written in chapter 10.

6.5 Iteration Assessment

6.5.1 Are the Goals Met?

The goal of this iteration was agree on User Interface design and build the system with 4 Batch View pages, 4 Insert – Edit pages and 1 View Details

page.

Because we were short in time (the system appeared to be bigger then in the beginning of the project), we did not succeed in implementing the whole system. We made proof of concepts – implemented at least a single page of each different type.

Chapter 7

Requirements

7.1 Introduction to Requirements

The main requirement, which our team had to perform during the project, was to copy the existing system from ASP to ASP.NET. The existing system was given us as a white box. We had a demo version of existing system. So our team was possible to try and test it, but no code was given. Indeed, the old system code would not be useful for us, because ASP.NET differs from ASP.

The second most important requirement for our team was the systems usability. It means that we had to put more attention on system user-friendliness. Our team expected that this requirement would be ensured by viewing the demo system, but later we admit it was not enough (look at section 7.3).

These two requirements were our main guidelines during all project and other upcoming requirements were relative with those two.

In the beginning of project (during first iteration) we admitted that it would be easier to manage requirements if we separate them in functional and non-functional requirements.

7.2 First Step - Domain Model

During the second iteration our team explored demo system and collected information required for domain model building. Since we knew the existing system not so well it was quite difficult to acknowledge all system's attributes and de-

tails, but finally we made it. After that we assembled all data together and produced the system's domain model (look at figure 7.1).

Figure 7.1: Domain Model

From Domain model we see the logical structure of the system.

Each project has many attributes, such as its name, size and start–end dates. Also, each project has a project manager, can have many events and many tasks.

An event example could be a steering committee meeting, or anything else, which is not a task, but occurs in a project life cycle. A task is an assignment for a specific employee.

Project manager together with Department manager are specializations of employees. The difference between ordinary employee and managers are:

1. Project manager can create projects
2. Project is possible to commit inserted tasks
3. Department managers are possible to commit inserted tasks, which belong to that department projects.

Ordinary employees can only insert tasks and edit them, create new projects (and become managers for that project).

A task is allocated in relation to a project for an employee. A tasks is characterized by specific set of attributes, such name, priority, begin – end dates and others.

7.3 One Important Mistake

We thought that it would be enough to collect requirements from demo system. In time we noticed that it is not enough to gather requirements only from given system, but it was also important to communicate closely with our customers.

Every week, usually on Wednesdays, we had meetings with our customers. In the beginning of the project during the meetings we just shortly presented what we have done without clear meeting content. Our customers were dissatisfied with that. So they provided some requirements for further meetings:

- There has to be one spokesman (one of team members), which would represent team's progress
- It is preferred to send meeting's program to the customers.
- The meeting's content have to cover three parts:
  1. What is done after last meeting
  2. What is projects progress at that time.
  3. What is estimated to do till next meeting
- There has to be a person, who would write customers answers to our questions

7.4 Use Cases

The next step was use cases development. They covered big part of functional requirements and described main work flow of developed system.

7.4.1 Purpose

Use Case specification is usually created as a part of system requirements. As the name implies, *Use Cases* are mainly intended to describe how users communicate with a system. As long as the main requirement for our system is usability, *Use Cases* are essential for us and requires exclusive attention.

Gathering *Use Cases*, it is very important to specify not only the the actual story how the user is using the system, but also to find out, how often it is performed. When designing a system, tasks which are performed most often should be simplified as much as possible, so they would take as less time as possible to complete them.

7.4.2 Collecting Use Cases

The process of collecting *Use Cases* was not very difficult for us, because most of them we could specify by observing an existing system.

Before collecting use cases, we made an use case template (all template is in appendix N.5.

Access Rights – in this section it is stated who can execute this use case.

Preconditions – in this section other conditions are stated which have to be full filled in order to be able to execute the task.

Postcondition – a description of changes, actions which happen if the use case is completed successfully.

Basic Flow – a successful story of how the user uses the system.

Task Insert

In our system *Task Insert* use case was emphasized among others. This case is very important for *Blue Bridge*, because project managers usually have to assign 10 or more tasks at once to team members.

In the old version of the system user has to insert tasks one-by-one: fill all fields for a task, submit it and then pick *Insert Task* again. Second drawback of the old system is that there is a framework, which creates insert window with

all possible fields. The problem is that in *Insert Task* some fields are irrelevant and has to be only in *Edit* mode.

In the new version, *Task Insert* window will look like an *Excel Sheet* and a user will be able to quickly navigate using *Tab* button between fields and fill in many tasks a time. Second improvement is that we were able to get rid of some unnecessary fields which were in the old version of the system.

7.5 Encountered Problems

This stage took longer. There was a couple of reasons for that:

- The team was not consistent of agreeing what actually constitutes a use case, and what style should it reflect, despite we had a template. Thus we spent too much time arguing how to do it than actually doing it.
- Our customer were busy, and we missed the time when they were ready to spend time with us.

7.6 User Navigation

To clearly understand the user navigation between pages, we made user navigation diagram (see figure 7.2).

Figure 7.2: UI Navigation

7.7 User Interface Requirements

This requirements gathering stage was mostly dependent from customers. Since our customers were IT systems Development Company, they understood and caught our mistakes well and gave us many suggestions.

To build user interface prototype we put more effort than we actually planned to. There was a coupel of reasons for that:

- In the beginning, we planned it to take one or two iterations. Actually, it took much more.
- User Interface prototype building was assigned for one team member. It was a difficult task for him, but other team members did not put sufficient effort to help him.
- Lack of experience in interface design area

7.8 System Prototype Requirements

This project development stage was planed to being implemented during 3rd iteration. This part of development process was parallel to UI prototype requirements gathering. It means we had spread our time in two parts to manage both works at one time. It was quite difficult, since UI interface required frequent reviews and meetings with customers. On the other hand we had also to collect system requirements and implement them. Fortunately, to collect requirements for system prototype it was quiet enough of demo system and earlier gathered requirements. There was not so big need to organize many meetings with customer.

During the 3rd iteration from domain model and collected use cases we already had even two suggestions for system prototype. Then we rebuild both of them and produce a new and better one, which covered both prototypes. the customers also got us some suggestions and soon after required changes they had agreed with our system prototype.

During the 4th iteration we build and test our system prototype. Though UI prototype was not finished at that time but it was not so much time left for divided work. So, during that iteration was decided to combine system

prototype with UI prototype and continue work when all project parts would be performed in one place.

At the beginning of 5th iteration the system prototype and UI prototype (though was implemented not all planed requirements) was finished and started all system development IN ONE.

7.9 Final Iteration Requirements

This iteration took most time in comparing with others. There was no need spread this iteration in two shorter ones, since our team knew most part of system requirements. We knew clear what we have to do:

- There was some holes in system security and rights providing
- The UI was also not so perfect
- The Real system users data were also not ported on our system
- Many smaller things, which was our headache

We also knew we have not so much time. Since all our situation was much clear now we were possible to predict how much time we will need. The predicted result was not such we expected in the beginning of the project: There was 160 hours lack. So it was made decision that it is better to port real data in the system with some mismatches than fix some small mismatches, holes and made UI very good.

Final result of our work was quiet good, I think, since we implemented the main part of our requirements (80 % of all). There left holes and mismatches, but from students point of view we got much expiriance and done wath we could.

The main lesson: “Experience is like path to somewhere, you can look back and see what you done wrong, but if you want go further you must put next step. No one knows if it be right step, but if you want learn walk you must go.”

Chapter 8

User Interface

8.1 User Interface

During the 3rd Iteration, we start creating User Interface prototype(UI). And the purpose of this section is to introduce with our User Interface prototype:

- Why we need UI prototype – 8.1.1
- UI elements and guidelines – 8.1.2
- Creation of UI prototype – 8.1.3
- UI prototype evaluation – 8.3.2

8.1.1 User Interface Goals

According to our customer, User Interface is most important part of this project.

The main idea to create User Interface is to give a look at the system and get the feedback, which would help to understand mistakes and bad design early enough to resolve the problems easily. The problems we can find with the help of User Interface prototype might be from choosing inappropriate colors for web, wrong navigation between pages to even omitted system functionality.

In the beginning we start studying how to create nice looking UI. The main task was to analyze Web Design elements.

8.1.2 Web Elements Functionality

It is one thing to create a set of web pages, it is another thing entirely to make a good looking, cohesive and well-designed set of web pages. To realize User Interface, we must know what Web design elements are and how to use them.

We start with collecting Web elements guidelines, study them and chose the best for our project. Down the page we show some guidelines, which are important in UI creation and realization.

Web Design Guidelines

1. Capitalize the first letter of only the first word of a list item, a list box item, check box labels, and radio button labels.

2. All Web pages should be structured for ease of comprehension. This includes putting items on the page in an order that reflects their relative importance. Pages should be long enough to adequately convey the information, but not so long that excessive scrolling becomes a problem.

3. Clearly differentiate navigation elements from one another, but group and place them in a consistent and easy to find place on each page.

4. There are several issues related to text characteristics that can help ensure a website communicates effectively with users:

   - Use familiar fonts that are at least 12-points;
   - Use black text on plain, high-contrast backgrounds;
   - Use background colors to help users understand the grouping of related information.
   - Use a familiar font to achieve the best possible reading speed.

5. In order to interact with a website, users usually require the use of screen-based controls (sometimes known as "widgets"). Besides the pervasive link, commonly used screen-based controls include pushbuttons, radio buttons, check boxes, drop-down lists and entry fields. Designers should ensure that they use familiar widgets in a conventional or commonly-used manner.

6. When pushbuttons are used, ensure that they look like pushbuttons and that they are clearly labeled.

7. Check boxes are used to select from among two or more mutually exclusive selections.

8. Radio buttons should be used to make binary choices, e.g., “yes”or “no”.

9. Drop-down lists are generally used to select one item from among many.

10. To speed user performance, show default values when appropriate, and do not limit the number of viewable list box options.

11. Entry fields are used when filling-out forms and entering text into search boxes. We should try to minimize the amount of information entered by users. Each entry field should be clearly and consistently labeled, with the labels placed close to the entry fields. To facilitate fast entry of information, designers should automatically place the cursor in the first data entry field, provide labels for each field (e.g., time, names, numbers, etc.), and provide auto-tabbing functionality. [1]

12. Coloration of the web pages can not be showy. Colours must be soft and in the page can not be striking contrast. It would be nicely if colours will not be very different. According to some colour-scheme standards, colours must go from dark to light or from light to dark. We have created few screen shots with different colours palette. They are needful, because can help to choose best variant for the last project version. Down the page we show few versions of different coloration. 8.2 8.1 8.3

After we have recognized Web Design elements, we start creating User Interface.

8.1.3 Building User Interface

To create UI we had two opportunities: draw it on papers or use web building program. We did not draw UI on papers because it would be difficult to use them and understand and trace the navigation between pages. However, it is

[1] http://usability.gov/pdfs/chapter15.pdf

Figure 8.1: ScreenShot1

Figure 8.2: ScreenShot2

Figure 8.3: ScreenShot5

not a bad solution, if there are only a few pages. But for us it was not a good idea, because as we have mentioned in the beginning of this section, it is very important to have all UI web pages. We chose "Visual studio" web building tool. This tool has all needful controls and in addition we could put various web elements onto the page - backgrounds, buttons, links, pictures, data grid, text fields, etc.

During User Interface creation cycle, we had few meetings with D. Masalas and A. Dienys. Furthermore, we received feedbacks and some consultative suggestions, also answers to questions about User Interface. Down the page, we show few examples from feedbacks and suggestions.

8.2 Cascading Style Sheets

During the Project Register system development user interface was one of most important requirements. It was necessary to make it nice looking and userfreandly. For this reason our team pay much attention on it. In third iteration our

team decides to use cascading style sheets. During the team meeting there were assumed CSS's advantages, estimated disadvantages and there were decided to choose cascading style sheets. There were some reasons for that:

1. There are provide only one CSS document.
2. CSS provide simple use and clear understanding, what styles, colors, alignments etc., are used in the system.
3. The System forms' design with CSS brought in some templates into our work.

8.3 CSS Evaluation

Finally our team had implement UI interface with CSS. It was clever decision, since our system's forms had to have similar look. This way to describe style was short, clear and efficient, it provide only one document through all system. CSS choice ensured light system style management and it was easy to change.

8.3.1 CSS Choice

Cascading Style Sheets must been studied during one week approximately. Thereafter it must been implemented. The chosen team member had collect required information related with CSS for about one week long. He also had spend some time by trying implement small CSS samples in ASP.NET. d some time by trying implement small CSS samples in ASP.NET.

8.3.2 User Interface Evaluation

Our User Interface was evaluated in Heuristic evaluation. A more descriptive term applied to heuristic evaluation is Usability Inspection: an inspection is carried out and a list of problems affecting usability is drawn up. Heuristic evaluation is a usability engineering method for finding the usability problems in a user interface design.

Here are some advantages for heuristic evaluation:

- It is cheap, because it is easy to understand, and use.

- It is intuitive and it is easy to motivate people to do it.
- It does not require advanced planning.
- It can be used early in the development process.

A disadvantage of the method is that it sometimes identifies usability problems without providing direct suggestions to solve them. The method is biased by the current mindset of the evaluators and normally does not generate breakthroughs in the evaluated design.

Different people can find different usability problems. In consequence of that reason, it is necessary to have few evaluators. In our project, our group members and the customer were evaluators and both we have found different problems and mistakes, which we will present soon.

We did heuristic evaluation in our project to identify possible flaws in User Interface (UI). The UI prototype was evaluated on the ten general principles for user interface design. For evaluation we have chosen few windows: Project View and Insert Task.

General Principles for User Interface Design

Visibility of system status *The system keeps users informed about what is going on* – there are clear presentment of information, so user might not be worry about non visible things.

Match between system and the real world *The system speaks the users' language, with words, phrases and concepts familiar to the user.* – All phrases and concepts are understandable and familiar to users.

Consistency and standards *Users should not have to wonder whether different words, situations, or actions mean the same thing.* – There are no such problem in our User Interface prototype. All different words, situations, or actions means different things (projektas, mano projektas, redaguoti, etc.).

Error prevention *"Even better than good error messages is a careful design which prevents a problem from occurring in the first place."*[2] – We do not used error message in UI prototype, but we will have them in our final system

[2] http://www.useit.com/papers/heuristic/

solution. In case to assurance we use error message, but it do not mean that we have bad design.

Recognition rather than recall *Minimize the user's memory load by making objects, actions, and options visible.* Instructions for use of the system is visible, so the user should not have to remember information from one part of the action to another.

Flexibility and efficiency of use *Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.*

Help and documentation *"Even though it is better if the system can be used without documentation".* – We decided not to use help and documentation in the system.

8.4 Conclusion

According to Web Design Elements and after evaluation, we have changed system design: labels, buttons, text boxes sizes and presentment. Also we pay attention to feedbacks and recommendations from customer: made error message in the system, use filtering, etc. Down the page, we will put few screen shots from the final User Interface prototype solution. Down the page, we will place few ScreenShots from our project. They will show, how was changed some details from the beginning to the end, and how look the system. 8.4 8.5 8.6

Figure 8.4: InsertEventsOld

Figure 8.5: Insert

Figure 8.6: insert New Project

Chapter 9

Implementation

9.1 Building Implementation Queue

Implementation process activities of the project had to face many various problems and obstacles. In the best-case scenario, the implementation activities are driven by requirements, collected and stored in one or another format. But of course, there are more things to it. Our project is short in time (approximately 600 human work hours). However, non of the team members has had any encounter with .NET platform. This put us in a state that we had to predict as many obstacles as possible and solve them.

Obstacles might be considered two-fold. One thing is that we do not know how to implement a specific method, or .Net does not support it transparently. Another obstacle is lack of experience, and experience is mostly needed in judging whether appropriate technique is at least sufficiently good to solve the problem.

9.1.1 Development Risks

The first thing we did to tackle these problems was to establish an order of biggest development risks and sort them, so we solve biggest risks first, thus defending ourselves from situation when spending half of the time 50% of work is done, but the next 50% is unpredictably harder to solve, if at least possible.

The predicted development risks where identified early in project, in "De-

velopment Risks" included in appendix K.

9.1.2 Use Case Prioritization

The complete list of use cases is defined is Use Case Specification document, included in appendix N.

During use case prioritization, we found that essentially there are only two types of use cases from the perspective of implementation: ones are when viewing many items in a list (UC4: View Projects, UC6: View Events, etc), another when editing or creating a new item (UC1: Insert Project, UC5: View Project In Detail, etc.).

Customers paid mostly attention to UC3: Insert Task, as this use case addressed very importantly usability issue - the old system was very unwieldy for inserting many tasks. Thus this use case was adapted so users could easily insert many tasks at once.

So, the team agreed, that for the sake of addressing architectural tasks first, we should build UC4: View Projects and UC3: Insert Task. The first use case is the main page, and the second one was stressed by customers.

9.1.3 Building Final Development Queue

Use case prioritization concluded with two use cases, which we should build first. However, aside from use cases, we had also to implement functional requirements.

Functional requirements are identified in Supplementary Specification included in appendix M.4. We identified three requirements:

- Multi-attachment ability.
- Filtering boxes in datagrid.
- Data Versioning / History.

The first requirement (Multi-attachment) was left for the second release. Filtering boxes requirement was prioritized as more important than Data versioning requirement. The team also agreed that History requirement affects all system, and must be done first, while filtering boxes can be postponed after.

Finally, we can present the list of development tasks we had to face during this whole project.

1. Develop Database Architecture
2. Understand Page Life Cycle
3. Build Architectural Prototype
4. Build Arklys Injections
5. Build UC5: Insert Task
6. Create DB Fillers
7. Rewrite Controllers
8. CSS
9. Understand Page Layout
10. Create PRControls
11. Build UC4: BatchProjectView
12. Build all the rest use cases.

9.2 Database Design

9.2.1 Problem

There is a requirement for the system to store data in a database and to log changes (M.4.1).

The requirement to log changes brings a lot of difficulties, because it requires more data to be stored which can lead to performance and data duplication issues.

9.2.2 Possible Solutions

Three different approaches were developed for logging changes:

Log a Field Change

Using this strategy a change of a field is logged storing its old and new values in an external database table.

Advantages:

- No data duplication. Since only the changed field are be logged no redundant data are being stored.

Disadvantages:

- Complicated design. Few solutions of this problem can be discussed but non of them seems to be good enough.
- Complicated usage. Tricky design leads to difficult usage.

Duplicating Entries

Using this strategy if a table row is changed it is duplicated regardless one or more fields are altered.

Advantages:

- Simple database design and implementation. This strategy brings less limitations for design and implementation comparing to others.

Disadvantages

- Performance. Usually only newest data a required and only sometimes old versions are examined. Because table rows are duplicated on every change, database grows very fast and queries can take more and more time which can lead to huge performance issues. This problem can be solved by having a identical table where only history are stored.
- Data duplication. Because on a field change whole role is duplicated, a lot of redundant data are stored in the database.

Split Tables

This strategy is similar to previous one. The difference is that all tables are divided into two. The first one holds data which does not change or changes

are not logged. In the second table data which has to be logged is stored. This table is managed the same like in the second strategy.

Advantages against previous strategy:

- Less data are stored.
- If implemented correctly, can make almost no affect for the performance.

Disadvantages against previous strategy:

- More difficult design. Since this approach brings some difficulties, they are not so big comparing to ones in the first strategy.

9.2.3 Our Choice

After analysing possible strategies it was decided to take the second one. It was considered the best, because it brings biggest advantages, least disadvantages, is easy to understand and implement.

9.3 The Page Life Cycle

The ASP.NET page life cycle concept is very important. The web programming must create and application, which appears for the user as a continuous running process. However, for programmers this is not the case (or at least was for ASP programmers prior ASP.NET). The server loads an ASP.NET page every time it is requested and then unloads it after the request is completed. Programmers before ASP.NET evolved had to restore the controls to their previous state manually (i.e. had to program it). ASP.NET has facilities which automates this process.

Although the communication between the client and the server is stateless and disconnected, ASP.NET provides means for programmers to use this in very transparent manner. The technology provides a set of events, which programmers use to achieve client experiencing a continuously executing process.

Figure 9.1 visualizes this set of events [1].

The page undergoes a total of eight stages.

[1] Diagram is taken from

0: Instantiation

1: Initialization

2: Load View State

3: Load Postback Data

4: Load

5: Raise PostBack Event

6: Save View State

7: Render

Figure 9.1: Page Life Cycle

Stage 0 - Instantiation

ASP.NET web page life begins with building a class hierarchy. This hierarchy is a represents an ASP.NET web page, which can be accessed and modified by code.

The first time the page is visited after a change to HTML markup or code belonging to that page, the server builds the class of that page. The class is inherited from System.Web.UI.Page class. This is done in either way, whether you write a code in-line in the aspx page together with html markup, or do it separately, that is linking aspx page to a code behind class, which in turn inherits the same System.Web.UI.Page class. The purpose of this autogenerated class is to programmatically create the page's control hierarchy.

When the hierarchy is constructed, all controls attributes are set to those defined in aspx file using HTML markup.

Stage 1 - Initialization

In this stage, every control fire its own and allows to do so for its children their Init event.

This events purpose is two-fold:

- Server controls don't begin tracking view state changes until right at the end of the initialization stage. This is done in the next stage.
- Second, when adding dynamic controls that need to utilize view state (to support the continuous feel of ongoing process), these controls will need to be added during the Page's Init event as opposed to the Load event (fourth stage), as we'll see shortly.

Stage 2 - Load View State

This state happens only when the page was posted back. It is done so, because the view state does not simply exist if the page was not posted back, but simply by a user accessing the page for first time. View state is created in stage 6 (after page events such onMouseClick or onButtonClick were processed).

Stage 3 - Load Postback Data

This state again happens only in case of post back.

In this stage, the page class enumerates all posted back data, and searched any corresponding controls. If it does find, and the control implements IPost-

BackDataHandler interface, it gives that data to the control. Then the control would update its view state (which finally will be saved in stage 6).

Stage 4 - Load

This event is fired when it is guaranteed that the page has been restored to its previous state.

Usually, programmers in this event load data to the controls from data sources.

Stage 5 - Raise PostBack Event

This stage raises various user events such as for DropDownList would be SelectedIndexChange, or for a Button would be a ButtonClicked event.

Stage 6 - Save View State

In the save view state stage, the Page class constructs the page's view state, which represents the state that must persist across postbacks. This field is saved in the page in a form of a hidden field.

Stage 7 - Render

The page is simply rendered by recursively rendering all controls. The rendered markup is returned to the server.

All these seven stages are very important to understand how to properly program business processes. We did omit some of the non-very-important stages such as Send postback change notifications, Prerender, Dispose and Unload.

9.4 Dynamic vs Static

One of the first decisions we had to make before starting building system was to choose in what way shall we build our web pages? There are at least two ways:

- Static building
- Dynamic building

The first one, the static building, is also known as drag-and-drop or What-You-See-Is-What-You-Get. This is when you just take already existing controls and drag and drop them on the web form, customize the appearance of controls, assign data and values, add links to other web pages and the page is almost ready to run.

The second approach is more complicated, since all the page is created programmaticaly. That is, for each page each control and each function must be created and altered programmaticaly.

Both ways are good, and using both ways we could have finished the system. However, the questions is which one to choose. And this is a good question, since:

Static building advantages: Static building is very good for building user interface prototypes. If you use the same technique for building a real system, then, possibly, we could transfer our user interface prototype web pages without any problems to our development system.

Static building disadvantages: Considerations with static building arise when we have to think of how we actually are to ensure the common page layout? If we are to build a system with 20 pages, will we have difficulties in making a slight change in a header, which appears in every and each page? Or if we have many pages with tables, how shall we ensure, that all tables have the same positioning in the page?

Dynamic building advantages: With dynamic building we gain the total control of the page - we create how we want it and we do it how we want. With static building, as for contrast, even event binding in some cases was not as a trivial task as it seemed from the first glance. With dynamic building, the problem of common page layout becomes a question of architecture - if we will be able to encapsulate all page building in some factory classes, we can afterwords alter factory products easily.

Dynamic building disadvantages: Dynamic building advantage is also a disadvantage - total control requires total understanding of the architecture, i.e. the effort to create code and create it so that it is extendable is a task worth experienced architecture designers.

Considerations for both: The biggest problem, however was the obstacle, which we had to face firmly and at length - we had completely no experience both for creating web applications, nor for the .Net architecture, nor c# language. For decent evaluation of static vs dynamic building techniques we had to acquaint with both of them. And this takes time.

9.5 Building two Functional Prototypes

So we had a problem of which way to choose for building UI, and this problem must have been solved very early, for this greatly affects all our future decisions. As shown above, we had a variety of ways to do so, and had a lot of information from web, but non of us actually was acquainted with any of these techniques, and no one could tell us firmly which way is the better.

Above all, we had no experience in .Net programming, so finally we decided that the best choice would be to build two prototypes and afterwords evaluate and choose to stick to one.

From the beginning, Martynas and Darius had separate ideas, and this resulted that: Darius will build his prototype "Nano Pages" with a mix of dynamic and static building, more focusing on dynamic contents and Martynas focused on building content visually and focusing on good code structure and reuse.

9.6 Darius Prototype – Nano Pages

9.6.1 Prototype Goals

A prototype was built as to reach these goals:

- User Interface can be built statically, but also allow means to alter the page on load.
- Separate page building, permission management, database access and visual design.

9.6.2 Explained

The system was divided into the following components: BBPageLoader, CodeBehind (CB), NanoBuilder and PermissionManager.

BBPageLoader acts as mediator, who controls the building process of web pages.

CodeBehind component classes together with .aspx files constitutes the web pages. NanoPageLoader intercepts the load routine of web pages, and then dictates the building process.

NanoBuilder, on request of BBPageLoader, "builds" the web pages. "Building" actually means modifying and configuring web pages, and not building those from scratch. This is done this way because the goal is to mix the dynamic building and visual design.

PermissionManager simply acts as a service provider. All it cares about is user identity and permissions.

To put everything in places, see the sequence diagram 9.2, of how the web page is built and how to act when events happen.

9.6.3 Class Diagram

Figure 9.3 shows the class diagram for such a design.

The diagram shows, and tries to explain, that:

- Each web page will have to pass the ownership of control flow to BBPageLoader by extending it.
- BBPageLoader must have access to PermissionManager and MasterBuilder. MasterBuilder will simply find which NanoBuilder sub-class is competent to build current page.
- Finally, for one Web Page, there can be many NanoBuilder classes, where each of them customizes the page as it likes. This might, and could have been used to build two NanoBuilder for Project web page. One NanoBuilder would build a project page witch would show all pages, the other one would show only projects, which are under one employees management (e.g. ViewMyProjects).

Figure 9.2: Sequence Diagram

Figure 9.3: Class Diagram

9.6.4 Advantages of this Prototype

- Divides a system into separate part, each part communicating with other parts through defined and established interface.
- Joins visual design and dynamic building.

9.6.5 Disadvantages of this Prototype

The main disadvantage was that this prototype lacked good file structure.

9.7 Martynas Prototype

9.7.1 Prototype Goals

The second prototype was build by Martynas. He considered these requirements to be most important:

- User interface must be designed visually (at least major part of it).
- Reduce duplication of code. It is very important, because *copy & paste* is the source of most bugs since you often forget to fix some places when refactoring.

9.7.2 Limitations of ASP.Net

Since he have got a lot of experience making windows style applications, he tried to use his knowledge creating this system as well. He made few attempts to create an architecture like for an ordinary application, but always failed.

He always failed because of limitations of ASP.NET, which are promised to be fixed in next release. The most irritating disadvantage of ASP.NET is that it is impossible to inherit web-pages if you are visually designing them. All the pages in the system has got similar layout and includes some common parts which naturally should be designed in a base page. Because of the mentioned drawback all the pages have to be designed from scratch and changed one-by-one if a change is needed.

Despite the limitations we stick to visual design because of the main requirement – usability. We must be able to reconfigure user interface often and quickly during development process if needed.

9.7.3 Design

The designing of a system usually starts by looking at it from different points of view (e.g. logical, data flow, user interface – Model View Control (MVC) pattern) and trying to distinguish different parts. Different parts means that they do different things and usually are not strongly connected and so can be developed separately.

In our system *Model* part (from *MVC*) is separated automatically, because *Data Base* is used to store persistent data. The Data Base design is discussed in section 9.2.

View

When looking to the user interface of our system, it is easily noticeable that there are a lot of similar pages which could be bound together. For example *Edit* is the same *Insert* page only filled with data which can be altered by a user. On the other hand, all the batch view pages differ only by data which is filled in, the layout is the same.

In my architecture he derived all the pages (in *ASP.Net* code behind can be inherited) from *BasePage* which does common tasks for all pages: checks whether user is logged, binds data on page load.

For every group of similar pages he made a base class which holds only main logic (without any user interface design) for particular group of pages:

- *BatchView*. This class is used when a page which shows a table with some kind of data is created (e.g. to show list of project). It is responsible to fill in data to a page, redirect to respective *Edit* page and so on.

- *Edit*. A page, derived from this class, can be either *Edit* or *Insert* page. The difference is that, if you are editing, the page is filled up with a data on load.

- *View.* This group of pages is intended to show particular data in detail. The *Edit* group can be used for this reason as well, but he separated them for GUI customization reasons.

Logics

There is not single package responsible for all the logics in the system. It is because that it completely differs when looking from different points of view. For example if you look at user interface, all the similar pages (see section above) have got common logics. So part of the logics is implemented in base class in user interface package.

On the other hand, if you analyse system from data point of view, you see that there are such parts:

- Projects,
- Tasks,
- Users,
- *etc.*

So he made another package which is responsible for communication between *User Interface* and *Data Base* packages. It loads needed data, filters and sorts it. New records are stored or existing ones altered using this package since it checks user rights and manages history information automatically.

9.7.4 Advantages of this Prototype

It clearly separated parts of the system which could be developed separately. Some code written in base classes in one place which prevented from some bugs.

9.7.5 Disadvantages of this Prototype

Later it was found out that classes and packages were not flexible enough and needed to be split into smaller ones. Classes which were responsible for retrieving and writing data into *Data Base* were separated.

During developing process it was found that pages in *User Interface* package has got a lot of differences even they look familiar. For example data in Batch

View pages has to be filtered in different ways (e.g by substring of a column or picking a value from possible ones using *Drop Down* list). So it is difficult to separate common logics for pages and base pages for groups turns out to be not very useful.

9.8 Prototype Choice

The two prototypes were finished in iteration 4.

We chose to stick to Martynas prototype because of his good structuring of files. Darius prototype maybe had system separation into different modules, but that was not reflected in the prototype on how the system was divided into physical files. This would have had a significant impact of how the team would work, thus Martynas prototype was considered as a better one.

Later, however, Martynas prototype was enhanced with features, that were lacked in it, but present in Nano Pages (see 9.10).

9.9 Building Arklys Injections

As a result of re-collecting use-cases with customers, customers exclaimed they would like to enter not one task at a time, but a variable count. This immediately raised an idea of dynamically adding edit boxes and buttons, which, of course, was still not an investigated area of .Net and c#. Thus Darius decided to try to build a proof-of-concept one single web page, were a user could dynamically add or remove buttons and this subsystem was called Arklys Injections.

The goal of Arklys Injections was to establish a common way to add an ability for every page to have dynamically created parts. This was achieved through the use of the following classes : **DynamicContentInjector and DynamicContent** and a web user control.

DynamicContentInjection is a controller class which provided all the programing interface to the user of the class.

DynamicContent was a class, which had to be inherited, in order to create a web user control. This web user control was used afterwords used by DynamicContentInjection to create as many instances as was requested by the

user.

Web User Control Web User Controls offer an easy way to partition and reuse common user interface (UI) functionality across ASP.NET Web applications. Like a Web Forms page, you can author these controls with any text editor, or develop them using code-behind classes. Also, like a Web Forms page, user controls are compiled when first requested and stored in server memory to reduce the response time for subsequent requests. Unlike pages, however, user controls cannot be requested independently; they must be included in a Web Forms page to work.

Web user controls were created statically using drag and drop. This control created the exact dynamic content and it's appearance which will be added to a web page.

9.9.1 How to Prepare the Page to Use DymacicContent

In order for the page to receive dynamic content, the DynamicContentInjection class had to be "*injected*" into the page life cycle.

1. Firstly, the DynamicContentInjection (DCI) had to be created passing the along the reference of asp page and the path to the newly created Web User Control (which inherits DynamicContent).

2. Secondly, the ASP.NET PlaceHolder control must be passed to the DCI (DynamicContentInjection) instance. This could have been done already in the building process of DCI, unless the creation is done not in the Asp pages constructor, since at that time the Place Holder is still not created.

3. In the page init cycle, the "Build" method of DCI must be called for the controlls to be built properly

4. Again in page init cycle, an event handler must be created to handle "Page_UnLoad" event, and that event must call "Save" method of DCI, to save the number and IDs of Web User Controls created.

These simple four steps prepares the page to create the dynamic content.

9.9.2 How to Add or Remove DynamicContent

To create or remove dynamic content is very easy, after you set up DynamicContentInjection. In order to add a control you must call *incrDynCont()* method on DynamicContentInjection object.

Similarly, if you want to remove, then call *removeControl(e.CommandArgument.ToString())*. The expression *e.CommandArgument.ToString()* produces the controls id you want to remove. Object e is of type *CommandEventArgs*, and this is passed along during event processing. Of course then, the code to remove the dynamic content must be placed in an event handler. To bind user code a remove click in dynamic content, you must bind an *onStartEventBind* event in init cycle. The total solution is shown below:

```
protected void onInit()
{
  ...
  DCI.onStartEventBind +=
    new CommandEventHandler(DCI\_onStartEventBind);
  ...
}

private void DCI\_onStartEventBind(object sender,
  CommandEventArgs e)
{
  DynamicContent dc = (DynamicContent)e.CommandArgument;
  dc.onRemoveClick +=
    new CommandEventHandler(dc\_onRemoveClick);
}

private void dc\_onRemoveClick(object sender,
  CommandEventArgs e)
{
  this.DynConInjector.
    removeControl(e.CommandArgument.ToString());
}
```

9.9.3 How to Create DynamicContent Web User Control

To create a DynamicContent Web User Control is no more difficult than creating a simple ordinary any other web user control. The difference comes only in two places: the web user control must inherit from *DynamicContent* class, and not directly from *System.Web.UI.UserControl*. And must override `public override System.Web.UI.WebControls.LinkButton lbRemovea`. What this means is that every DynamicContent we create will have to have a remove *LinkButton*.

9.9.4 Problems Creating Arklys Injections

Arklys Injections was thought out during one night, but implementing it required three days, mainly because of lack experience in .Net.

Delayed DynamicContent Creation

The first problem we ran into was the delayed reaction of DynamicContentInjection (DCI) to user click. The problem was, that if the user click the button to add a dynamic content, the event processing user request was executed after page's init cycle was over. This meant that at first the dynamic controls are created, and only then the DCI received a message, that user wants one more DynamicContent. So the results was that if the user wanted to see 5 controls, he would see only 4. If he wanted to see 1, he would see zero. And so on.

We tried to overcome this problem by passing the number of DynamicControls to be created using a HyperLink, like this:
`(HyperLink)hl.NavigateURL="insertTask.aspx?DynamicContent=ContentNumber+1"`.
This solution, of course, did not work either, as by using HyperLinks the page was not submitted back, but rather redirected to itself. This way, the .Net did not executed very important .Net page life cycle phases, namely SavePageState and LoadPageState. The results was that the controls were created at correct phase, but the state of the controls was not restored. That is, if we write something say, in a text box, that information is not persisted when we add another DynamicContent. Thus another workaround had to be thought.

The solution, though, was lamentably simple: as soon as DCI receives re-

quest for new DynamicContent, it can create it and it will appear as expected. Then whenever the DCI recreated the DynamicContent, it will do so in the page's init cycle, and thus the controls view state information will be retained and persisted correctly.

Correctly Removing DynamicContent

As was told before, to remove the DynamicContent is as simple as calling `DCI.removeControl();` and pass and ID of an appropriate control. However, for us to code the class was not as simple as now it is to use it.

The problem soon occurred when we tried to remove a DynamicControl (DC). Imagine a situation were we have three dynamic controls. Each of them have a text box. Let's say these values are contained in each of those DCs:

first DC: id=0. TextBox.Text="aaaaa".

second DC: id=1. TextBox.Text="bbbbb".

third DC: id=2. TextBox.Text="ccccc".

As we know already, .Net restores view state by looking at controls ID. And as we create programmatically DynamicContent, we have to assign IDs to controls also dynamically. The easiest way to do it is to assign numbers as ID, and constantly increment the number, that is `0, 1, 2, 3` and so on.

Now image what would happen if we remove and then add a control. Firstly, when we click the button to remove control, all controls are restored, and then the remove event is called, where we remove the control. Let's say we remove the control which `id=1`. What we have now is two DC with the following values.

first DC: id=0. TextBox.Text="aaaaa".

second DC: id=2. TextBox.Text="ccccc".

Now, we click a button to add a third control (since we have only two now). The first thing happens is DCI is recreating the DynamicControls. However, what we get is one control with `id=0` is recreated normally, while the second one gets an id of 1 instead of 2, and thus the information in the text box is lost.

first DC: id=0. TextBox.Text="aaaaa".

second DC: id=1. TextBox.Text=“”.

When the third control is added, it gets an id of 3, and text box will also be empty.

The idea to solve a problem was to somehow persist the IDs of our DynamicContent controls. The solution was though soon and coded in 5 or 6 hours. It introduced a new step in Arklys Injections life cycle - the PageUnload stage had to save the IDs of each Dynamic Controls into SessionState, so it could persist through many page requests. This is the reason why we have to bind to Page_Unload the following event:

```
protected virtual void InsertView_Unload(object sender,
  EventArgs e)
{
  DCI.save();
}
```

9.10 Improvements to System

9.10.1 New DBFillers and Improved Controllers

Controllers

As the system grew, the old controller classes began to not only not fulfill their role, but also became a hindrance to further enhancing our system. We had to redesign them.

The new role for controllers was only to perform data modification in database routines, such as inserting new rows, updating or deleting old ones. The new role for controllers was to be an intermediate layer abstracting access to DB. This abstraction suited perfectly for data-history requirement.

This requirement stated, that every data change must be stored in DB for later use and that this requirement will be implemented later, after the system will be ported to .Net. So, this new role for controllers allowed us to port to .Net without worrying that later this requirement till have big impact to our system.

DB Fillers

If controller classes now cared only about data modification, then who had a role of getting data from the database? This new role was decided to be performed by DataBase fillers. These fillers gathered data from DB and filled the appropriate structures.

9.10.2 FilterDropDownList

To implement data filtering successfully, we had to create our own custom web controls.

The new custom web control was named simply FilterDropDownList. This control was an extension to a normal drop down list, except it had a few new attributes: FilterOperation, FilterColumnName, DefaultValueID and DefaultValueText. The first two attributes were used to build a filter rule, by witch the rows where filtered. The two latter attributes created a default item with a default value.

9.10.3 FilterTextBox and DatePicker

Later on, we also created two other controls. One was like a standard text box control, except it had FilterOperation and FilterColumnName. The other was an extension to a calendar. The Calendar control is used to display a calendar in the browser.

9.11 Final System Design

9.11.1 Package View

The final system design resulted in the following packages:

- User Interface. This package holds all the logics (e.g. navigation between windows) and user interface part.
- DBFillers. This package extracts data from database.
- Controllers. This package commits data to database, ensuring any data relation requirements.

- User. This package is repsonisble for permission control.
- PRControls. This package is an extension to .NET user interface controls

The packages are visualized in the 9.4 diagram.

Figure 9.4: System packages

User Interface

All the pages must be derived from the *BasePage*. This class is responsible just for few base functions, such as controlling the life cycle of a web page.

For every group of similar pages, a base class is made which holds only main logic (without any user interface design):

- *BatchView*. This class is used when a page which shows a table with some kind of data (e.g. to show list of project).It is responsible to fill in data to a page, redirect to respective *Edit* page and so on.

- *Edit*. A page, derived from this class, can be either *Edit* or *Insert* page.
- *View*. This group of pages allows to view a particular data in detail. The *Edit* group can be used for this reason as well, but they are separated for GUI customization reasons.

An example of how a Batch Project view is composed of classes can be seen in 9.5 figure. It shows that this page inherits BatchView page, and BatchView inherits BasePage. Through dependiencies, batch project view page can acess FilterDropDownList custom web control, and abstract controller for data access.

Figure 9.5: Batch Project View class diagram

The total inheritance diagram of BasePage class can be seen in 9.6 diagram.

DBFillers

DBFillers is responsible for:

- Retrieving required data for *User Interface* package;

There is an abstract base class *DBFiller*, which defines an interface. Each subclass of a *DBFiller* will fill a dataset with some data from the table.

Figure 9.6: Base Page inheritance

The class diagram with all sub-children can be seen in figure 9.7.

Controllers

This package controls data submission to databse actions, ensuring History requirement is satisfied and hiding history implementation features from programmer.

There is an abstract base class *AbstractController* which defines an interface and performs most common tasks. All base classes in the *User Interface* package work with data using an instance of this class.

The class diagram with all sub-children can be seen in figure 9.8.

PRControls

This package provides new or enhanced web controls for user interface design. And example of these would be FilterDropDownList or FilterTextBox.

These controls are used only by User Interface package.

9.11.2 Project File Structure

All our system was finally divided into systems. All these systems were visual studio projects:

- Project Registry. It is a main project, which actually builds the system.

Figure 9.7: DBFillers inheritance

Figure 9.8: Abstract Controller inheritance

- PRControls. This project builds additional web controls, such as Date Picker of Filter Drop Down List.
- PRCommonLibrary. This is a project, which holds common code used by other two projects.
- UIPrototype. This project is for building User Interface Prototype.
- PRTest. This projects performs unit tests, and requires NUnit package.

Project Registry

. This project actually build the ASP.NET web application. It uses both PRControls and PRCommonLibrary projects.

The following list describes projects file structure in detail. Each item is a directory. Each indented item denotes a subdirectory.

ProjectRegistry: a directory to hold ProjectRegistry project

- **Database:** Each controller is put into separate directory. Examples: Employee, Event, Project, etc.
  - **Backup:** holds a backup of database, which can be used to restore or create a new database.
  - **Datasets:** Each subdirectory holds a DBFiller, which fills a dataset with data. Examples are: Departments, Employees, Years, ProjectManagers, etc.
- **HeaderControl:** Holds Header and Copyright user controls.
- **JScripts:** Holds any java script codes. Example: datepicker.
- **Pages:** Holds groups of sub-pages
  - **Administrator:** Holds administrator pages (Administrate)
  - **BatchViews:** Holds batch view related pages (BatchEmployeeView, BatchEventView, BatchProjectView, BatchTaskView)
  - **DetailView:** Holds detail view pages (ViewProject)
  - **Inserts:** Holds insert/edit pages (InsertEmployee,Insert Event, InsertProject, InsertTask, etc).
- **Pictures:** Holds pictures

Scripts: Holds this project solution file

Users: Holds user package

PRControls

This project builds various web custom controls.

Projects file structure: The following list describes projects file structure in detail. Each item is a directory.

PRControls: a directory to hold PRControls project. Holds web custom control files (DatePicker, FilterDropDownList, FilterTextBox)

PRCommonLibrary

This project contains source code used by other projects.

Projects file structure: The following list describes projects file structure in detail. Each item is a directory.

PRCommonLibrary: a directory to hold PRCommonLibrary project.

Persistency: Holds Persistency classes (FilterHashTable, SessionPersistency)

UIPrototype

This project builds user interface prototype.

It is built simply by dragging and droping controls.

PRTests

This project runs unit tests to test ProjectRegistry project.

It uses NUnit http://www.nunit.org/

Chapter 10

Testing

10.1 Introduction

Our testing process was done according to guidelines stated in document "Test Plan" Q. It was planned to have such tests:

- functional – ensure proper functionality, including navigation, data entry, processing, and retrieval.
- user interface – ensure that the User Interface provides the user with the appropriate access and navigation through the functions and all the pages have a common style.
- security – ensure that only those users granted access to the system are capable of accessing the system.

Security testing has not been done, because only few security features has been implemented till the end of the project.

10.2 Purpose of Testing

Testing is intended for checking whether a system operates as it is expected. The size of testing process varies depending on a project, but it should be started early in the project and performed till the end of the project. The sooner a bug is found, the easier it is to fix it.

> "Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence" Edsger Wybe Dijkstra.

10.3 Test Types

Possible testing methods varies depending on an area you want to check. For example there is no other way to identify usability problems than asking people to use a system and give a feedback. This kind of method should be avoided if possible, because it has got a lot of drawbacks:

- Since labor is expensive, it rises development costs,
- This method is very slow, so it takes a while to check all the system,
- Humans do a lot of mistakes, so a few people should test same things in order to minimize a chance of a mistake.

For parts of a system where it is possible to check correctness of a functionality by writing a simple condition, automated tests suits best. The biggest advantage of automated tests is that you can check a system very fast. Despite that automated tests avoid all the disadvantages inherent in manual testing, unit test have got drawbacks too:

- Lacks flexibility, so they must be adjusted whenever system architecture is changed a bit,
- Tests must be written for every single part for a system. However, writing tests is a big issue, since many programmers dislike doing that.

During the development process we used both types of tests for measuring and sustaining quality of the system.

10.3.1 Unit Testing

As the name implies, this test type is designed for doing automated tests for units of a system. Unit testing is one of the key things of Extreme Programming. It suggest writing test cases before writing code. At the begging it could seem to be an overhead, but in the reality you get more advantages than disadvantages:

- even a simplest test prevents you from a lot of bugs,
- in order to write a test, your code has to be well structured,
- bugs do not return or they are located faster,
- if you have enough tests, you can refactor code without worrying about the impact on the other parts of a system, because tests will do it for you.

So if you are using unit tests in a project, refactoring becomes painless. This is important, because usually it is almost impossible to make perfect initial system architecture, because humans do mistakes and requirements tend change. In case you have to *refactor* a part of a system, you must also check how changes affect the rest of it. Good unit tests help to find mis-functional parts quicker.

Despite we had chosen RUP as a methodology for development process, we could not avoid code refactoring. First of all, this was because the client often provided us with feedback which usually brought changes. On the other hand we lacked .Net programming experience and needed to check several solutions in order to find the most suitable one. In the beginning constant refactoring often resulted in non functional system which bothered other team members. After implementing unit tests, a developer could easily check if everything worked correctly before committing changes to the main repository.

Unit Test Examples

It was decided to make a unit test for every page in the system checking if it loads correctly. It was efficient enough, because other parts of the system which are needed to load a page are tested automatically, and did not take much effort.

An example test checking if the main page loads:

```
using System;
using NUnit.Extensions.Asp;
using NUnit.Extensions.Asp.AspTester;

namespace PRTests
{
```

```
  public class PageExistenceTests : WebFormTestCase
  {

    string baseUrl = "http://localhost/ProjectRegistry/Pages/";

    /// <summary>
    /// test checking if "BatchProjectView" page loads
    /// </summary>
    public void TestBatchProjectViewPageExistence()
    {
      Browser.GetPage(baseUrl +
          "BatchViews/BatchProjectView.aspx");
    }

    /// <summary>
    /// test checking if Project Insert and Edit pages load
    /// </summary>
    public void TestInsertProjectPageExistence()
    {
      Browser.GetPage(baseUrl +
          "Inserts/InsertProject.aspx");
      Browser.GetPage(baseUrl +
          "Inserts/InsertProject.aspx?ProjectID=1");
    }

  }
}
```

10.3.2 User Interface Testing

User Interface testing is expected to identify look (e.g. colors, fonts) & feel (i.e. usability) problems. These tests are very important in our project since system is intended to be actively used by people in their daily work and usability was

stressed by Blue Bridge as a key requirement.

User Interface can be tested only by humans, because only users can tell whether interface is convenient for them or not. In our project we did two stage interface evaluation. First of all we tried to make it comfortable from our point of view and then Blue Bridge provided us with feedbacks. Since the project was intended to be used only internally, such testing was sufficient for us.

For further details on User Interface and its testing see chapter 8.1.

Appendix A

Iteration 1 – 17 08 - 22 08 Inception Phase

A.1 Goals

1. Agree On Development Case
2. Initiate project

A.2 Objectives

- Agree on Development case.
- Define Deliverables with templates and guidelines
- Establish following documents:
    1. Risk List
    2. Risk Management Plan
    3. Software Development Plan
    4. Use Case Model
    5. Supplementary Specification
    6. Team Contract

7. Test Plan

- Define Phase Plan
- Define Project Organization
- Create Programming Guidelines
- Create Domain Model

A.3 Work Orders

1. Martynas
    - Establish Risk List, Risk Management Plan
    - Supplementary Specification
    - Coding Standards
2. Viktoras
    - Establish Use Case Model
    - Test Plan
    - Create Domain Model
3. Darius
    - Establish Software Development Plan
    - Team Contract
    - Phase Plan
    - Iteration Management Plan

A.4 Iteration Assessment

A.4.1 IterationSummary

Evaluation Date

Iteration Assessment was carried out on 22 08 2005.

Concise Summary

This iteration resulted in project initiation. The primary objectives of agreeing on methodology and building a development case have been met as a result producing a development case.

Summary of Meeting with the Company

No meeting with the company.

Summary of Meeting with Project Adviser

No meeting with the project adviser.

Appendix B

Iteration 2 – 23 08 - 30 08 Elaboration Phase

B.1 Goals

1. Establish a good base for requirements
2. Build an initial prototype

B.2 Objectives

See table B.1 for objectives.

B.3 Iteration Assessment

B.3.1 Iteration Summary

Evaluation Date

Iteration Assessment was carried out on 24 08 2005

Summary of Meeting with The Company

Date:24 08 2005 Requests for system:

Objective	Worker
Create Software Architecture Plan	Darius Damalakas
Domain Model	Viktoras Čiumanovas , Vilma Rudžionytė
Collect Use Cases	Vilma Rudžionytė , Viktoras Čiumanovas
Make initial DB schema	Martynas Kriaučiūnas
Prioritize Use Cases	Darius Damalakas
Build a small prototype from UC1 Create New Project N.6, and UC3 Edit Project N.8	Darius Damalakas , Martynas Kriaučiūnas , Viktoras Čiumanovas
Prototype must connect to database	Darius Damalakas , Martynas Kriaučiūnas , Viktoras Čiumanovas
Refine Risk List	Darius Damalakas

Table B.1: Iteration 2 Objectives

1. Ability to reserve projects.

   The problem arises when a project manager tries to "reserve" an employee for a non-started project. If the project is approved, and project starts, a reserved employee continues to work for that project and no problem occurs. If the project is canceled, problem occurs if the project is forgotten to be deleted from the system. In this case, the resources are not freed up.

2. Load report (uzimtumo ataskaita) should not report abnormal resource usage, such as one employee working 150% of total time.

3. Ability to attach more than one attachment, for both projects and events.

Team Report For Customers:

1. The team should have a **meeting-leader**. This way, the order of meeting will be retained. Meeting leader is responsible for ensuring everything is ready for procedure.

2. **Presentation plan** must establish a defined order of meeting.

3. Presentation plan must include **a questionary**, and answers to these questions must be written down and well agreed both between team and customer. Questionary must be prepared before the meeting and reviewed by team.

4. Customers are interested to know which point in the project are we in at the moment in terms of accomplished jobs. If any important jobs are unfinished, what caused the delay. Almost finished jobs are not finished.

Date: 26 08 2005

- Do we have to implement Administration feature in the system during first construction phase? *YES.*

- Do we have to implement various reports (Ataskaitos) in the system during the first construction phase? *NO, not necessarily.*

- Do we have to implement multi files attachment feature during the first construction phase? *NO, not necessarily.*

- Is it preferable, that for every attached file would be possible to specify a file type? *YES. It must be a list of structured titles, such "Agreement", "Analysis Result", etc.*

- Who has the right to see information in Administration area? Administrators. *The reason why you (the team) can not access that part is a system bug. This must be solved.*

- Who has the right to edit and delete Project information? *Project Manager and Department Manager, who is in charge of the project. If the project's manager or person responsible for task changes, he/she loses the right to manage it.*

- Do we need to create individual table with "My Events" information as is done with "My Projects" and "My Tasks"? *No, it is not needed, since there are not so many events.*

- Who can perform all actions with events? *Project Manager, Department Manager and Employee who creates an event.*

- Who can perform all actions with tasks? *Project Manager, Department Manager and Employee who creates a task.*
- In Administration− >Personal Workers Information is "Work" column. Who manages this column: Administrator or the system counts it dynamically ? *Administrator. It is important value, by which system knows to show worker in employee lists or not.*

Summary of Meeting with Project Adviser

Meeting among:

Team members: Martynas Kriaučiūnas , Darius Damalakas , Viktoras Čiumanovas and

Project reviser: Michael Claudius.

Given suggestions:

1. The section "Team Contract" should be called "Team Members",
2. The section "Team Capabilities" should be called "Human Resources",
3. It have to be an explanation, why each team member get particular role(-s) or responsibility(-ies),
4. It have to be person who is responsible for documentation management (Documentation manager).

Appendix C

Iteration 3 – 31 08 - 13 09

C.1 Goals

- Functional Prototype
- User Interface

C.2 Objectives

See table C.2 for objectives.

C.3 Iteration Assessment

C.3.1 IterationSummary

Evaluation Date

Iteration Assessment was carried out on 14 09 2005.

Concise Summary

The goals were to build user interface prototype and functional prototype. The iteration was very unsuccessful, and we did not finish any of the defined tasks.

User interface prototype only began to shape, and use cases were not done up to the end as we wanted to, however, a lot of input was done to this activity.

Functional prototype area the work was very concentrated on building two prototypes. However, the outcome of the prototypes is good - the team agreed on how the system will be further developed, and what way it will focus on - that is it will stick to building the system with drag and drop techniques.

Summary of Meeting with The Company

Secret Meeting. Participants: . Dalius Masalas and Darius Damalakas

Date: 01 09 2005

Summary: Reviewed our Use Case Specification and wrote comments. Main errors : no *Glossary* results in inconsistencies between terms (such as "Project Owner", "Leader" And "Project Manager"). Need careful attention to pre- and post- conditions of every use case. Also some small (though sever) mistakes in use case flow (UC3).

Reviewed UI Navigation diagrams. Missing one page. Emerges a new need for a component diagram, i.e. what components builds what pages. This UI Navigation shows what the user perceives, and not how the pages will be implemented.

Minor error in Use Case Diagram.

Discussed a possible page design technique (with Master builder and Nano-builders).

Meetign date: 2005 09 07

During the project presentation:

- We must clear defined what is a finished product, when we speak about final result.
- All projects', tasks' and events' in the system can not be simply deleted – those data have to be stored in history area.
- Do we have to implement the Back button in the system, since it is implemented in browser too? It is not necessary. If you be able to ensure good system functionality without Back button you may skip it.
- Should we propose an additional document for interface prototype or it will be enough to show some screen shots? *YES, you may do that.*

Objective	Worker	Needed hours and date
Complete a **User Interface Specification**: for each of 19 pages build a window with visual studio and insert the screen shot of that page into UI specification	Viktoras, Vilma	There are total 19 pages. Initial page will take 4 hours of work, each subsequent will take 30 minutes, thus total amounting to 4+18*0.5 = **13 hours of work**. less than 4 days of work (4 hours per day). Due date: **09 09 2005**
Complete a **Test Plan**. Test plan must state what strategies will be used for testing, what are the requirements of testing, the priorities and risks for tests, acceptance criteria. Identify, evaluate and plan test deliverables, including test cases, test procedures and test scripts	Darius, Vilma	Initial investigation into test plan - 2hours. Assess test strategies (around 7 strategies) - 7*0.5=3.5 hours. Identify requirements - 2 hours. Assess and priorities tests - 1 hour. Build a test plan with scheduled deliverables - 2 hours. Total **10,5 hours**. 3 days of work. Due date: **09 09 2005**
Build a Database Package. Package will contain classes to group database access into one area by specifying concrete interface. This will abstract our access to database, hiding any additional functions, such as data versioning.	Martynas	Due date **09 09 2005**
Build a SAD. This includes generating DB schema, producing class diagrams, system sequence diagrams, packages diagram and writing programming guidelines (such as how to use DB).	Darius, Martynas	DB Schema 4 hours. Class Diagram 2 hours. Sequence diagram 3 hours. Package diagram 2 hours. programming guidelines 2 hours. Total **13 hours**. Due date: **13 09 2005**

Table C.1: Iteration 3 Objectives (part 1)

Objective	Worker	Needed hours and date
Build DB scripts. DB scripts should create all tables, add all constraints and all relationships into database.	Martynas	Total **2 hours** Due date **13 09 2005**
Build Use Case Specifications. This includes defining and detailing all currently known 17 use cases. Use cases must be of deliverable quality to the customer. Any important vocabulary (such as who is employee and who is project manager) must be defined in glossary	Martynas, Viktoras	Use cases are almost in detail, they need only a thorough review and to add vocabulary into glossary. Total **2 hours**. Due date **07 09 2005**
Build **Core package**, **Permission**, **ProjectBuilder** This includes building a NanoBuilder, NanoPageLoader, BBPage classes, CBProjectDetail and NanoProjectBuilder and NanoMyProjectBuilder	Darius	The actual work will be done in connection with other packages, mainly Database Package, Permission and Project builder. Difficult to assume the total amount of hours needed, but lets say each class takes 1 hour to build (total 6 hours), and then 3 hours to integrate those classes. To test and ensure that all packages work as expected 2 hours. If any problems occur on architectural scale, the time needed to fix this is undetermined. Total **11 + x hours**. Due date **16 09 2005**

Table C.2: Iteration 3 Objectives (part 2)

Appendix D

Iteration 4 – 14 09 - 28 09

D.1 Goals

1. To create best Use case solution and finish with them.
2. Establish the following documents:
   - Data Base architecture
   - System architecture
   - Test Plan
   - Use Case
3. Transfer partial User Interface on the system (shift, align, etc) and to run system prototype for the clients and other users.

D.2 Objectives

See table D.1 for objectives.

D.3 Iteration Assessment

D.3.1 IterationSummary

Evaluation Date

Iteration Assessment was carried out on 29 09 2005.

To make Data Base architecture documents	Martynas Kriaučiūnas
System architecture	Martynas Kriaučiūnas , Viktoras Čiumanovas
Make Test plan	Viktoras Čiumanovas and Martynas Kriaučiūnas
Finish Use case and make documentation	Darius Damalakas , Vilma Rudžionytė
Continue System core programing	Martynas Kriaučiūnas and Darius Damalakas
Make User interface	Viktoras Čiumanovas and Vilma Rudžionytė
Make Initial prototype for testing	Martynas Kriaučiūnas

Table D.1: Iteration 4 Objectives

Concise Summary

The iteration again does not meet its defined outcomes. However, of the defined goals some are done, , that is system architecture and database architecture. The two are discussed by team and defined formally (L.3).

Requirements are undergoing hard-times as before. The team is having meeting with customers, but the requirements are not formally captured. User interface prototype already contains 5 out of 20 pages, and they are already agreed and approved by customer.

Test plan is not even started.

Summary of meeting with the company

Meting date: 2005 09 14

Meeting purpose: Proposed Use Case document, User Interface planning.

- It is preferable to implement the Task insertion form like line (something like in Excel).

- The Project's name dropdown list can be implemented singly from the others insertion fields, since it suits usually for all task during one insertion.
- The User must have possibility to choose how many Task insertion fields he/she would like to see (i.e. 10, 15, 20 etc.).

Meeting Date: 2005 09 21

Meeting purpose: User Interface prototype review and Use Case document discussion.

- The Task column in Projects table can be some images, since it would save some place and would be more understandable.
- Team members must add two new columns into Projects' table: Defined project hours and used project hours. Those columns must be placed after End date columns.
- It is better to implement the column Project finished with words (Yes/No) or some images, instead with checkboxes.
- The Priority column is not required in Task insertion table.
- The task Improvement description field is not required in Task insertion table.
- The Column header fields, which contain time data, from the beginning must be empty.
- From the beginning of data insertion the first field in Task insertion form must be focused.
- It is enough, that the Description field would be two rows high.
- It is preferable to have the same task view and edit form (edition function would be allowed just for those who has required rights).
- It is preferable, that in messages about some changes in projects, tasks and etc., for which they are responsible, would be clear what changes were made.

Summary of Meeting with Project Adviser

No meeting with project adviser.

Appendix E

Iteration 5 – 29 09 - 28 10 Construction Phase

In iteration 5 the planned was remade for the last time and we committed to do the job till the end of our project.

E.1 Iteration Goals and Objectives

1. Implement system,
2. Improve User Interface prototype,
3. Write project report.

Implement Task Insert page	Darius Damalakas
Implement filtering	Martynas Kriaučiūnas and Darius Damalakas
Make User Interface prototype	Viktoras Čiumanovas and Vilma Rudžionytė

Table E.1: Iteration 5 Objectives

E.2 Requirements and UI Prototype

According to our customer, User Interface is most important part of this project. So at this iteration a lot of work was done making and tuning user interface.

The highest priority was given to Insert Task page, because it was used very often and was inconvenient. The requirement was to make task inserting as fast as possible. So the page was made similar to a *Microsoft Excel Spread Sheet* and it was made that few tasks could be inserted at once.

In this iteration was agreed on a common design of *Batch View* and *Insert/Edit/View* pages.

All the designing was made using CSS (*Cascading Style Sheet*) technology, so it was easily applied on the real system.

More about User Interface design is written in chapter 8.1.

E.3 Development

In the last iteration system design was made and the development was started using it. During the development process we realised that system architecture was not good enough and it was changed.

E.3.1 Database Subsystem

First of all *Database Controllers* were split into *Database Fillers* and *Database Controllers*. The task for *Database Fillers* was only to fill a table with data. The need of them came up when we realized that few *Database Controllers (old ones)* need to fill in the same data (e.g. most pages required data about projects), so we decided to reduce duplication of code by extracting parts of it into *Database Fillers*. *Database Controllers* made was responsible only for inserting new records and updating existing ones.

E.3.2 Filtering

When designing User Interface prototype, there was a request to add filtering fields above each column and it was required that filters would work while navigating through pages.

Filters influenced system architecture as well, because it system had to load filters on page load and gather data according to them.

It was decided to store filtering rules in browsing *Session*. To implement data filtering successfully, we had to create our own custom web controls:

- *FilterDropDownList* – This control was an extension to a normal drop down list, except it had a few new attributes: FilterOperation, FilterColumnName, DefaultValueID and DefaultValueText.
- *FilterTextBox* – This control was an extension to a normal text box, except it had a few new attributes: FilterOperation, FilterColumnName.

E.4 Testing

In this iteration testing was started.

Our testing process was done according to guidelines stated in document "Test Plan" Q. It was planned to have such tests:

- functional – ensure proper functionality, including navigation, data entry, processing, and retrieval.
- user interface – ensure that the User Interface provides the user with the appropriate access and navigation through the functions and all the pages have a common style.
- security – ensure that only those users granted access to the system are capable of accessing the system.

Security testing has not been done, because only few security features has been implemented till the end of the project.

In the beginning constant refactoring often resulted in non functional system which bothered other team members. After implementing unit tests, a developer could easily check if everything worked correctly before committing changes to the main repository.

We benefited a lot from testing, because bugs were caught faster and we were always sure that the system is always well functioning.

More about testing is written in chapter 10.

E.5 Iteration Assessment

E.5.1 Are the Goals Met?

The goal of this iteration was agree on User Interface design and build the system with 4 Batch View pages, 4 Insert – Edit pages and 1 View Details page.

Because we were short in time (the system appeared to be bigger then in the beginning of the project), we did not succeed in implementing the whole system. We made proof of concepts – implemented at least a single page of each different type.

Appendix F

Iteration Management Plan

F.1 Introduction

The goal of this document is to specify how to:

- Create new iteration plan
- When and how to evaluate iterations

F.2 Document Holder

Darius Damalakas is responsible for managing this document.

F.3 Iteration Timing

Each iteration is agreed by all team members to last approximately one week. Each iteration is evaluated on Tuesdays. Every new iteration starts on Wednesday.

On Tuesdays team evaluates iteration, and afterwords *together* agrees on next iteration plan.

On Wednesday, iteration can are re-evaluated and adjusted as necessary.

F.4 Iteration Plan Structure

Each iteration plan must clearly state:

1. When iteration *starts* and *ends*.

2. *Goals*

   Goals define the broad-level targets of this iteration. An example of such might be "Detail 2-3 use cases" or "Build prototype".

3. *Objectives*

   Objectives are concrete targets of this iteration. Objectives, in the best case scenario, must be specified as concrete as possible, and, if possible, in a way so they can be measured or tracked. An example might be "Detail a use case named. . . . . " or

   "Implement that and that component. Component realisation must result in the following artifacts:

   - Source Code
   - Updated Design Model
   - Updates Class Diagram

   ".

F.5 Iteration Evaluation and Assessment

Iteration evaluation is carried out orally by *all team members*. Evaluation process consists of the following process:

- Checking are the goals met
- Identifying succeeded and failed objectives
- Reevaluating risks
- Planning next iteration
- Shortly summarising evaluation process for later use using the following template. Attach this summary to iteration plan.

- Evaluation date
- Concise summary
- Summary of meeting with the company
- Summary of meeting with project adviser

Appendix G

Software Development Plan

G.1 Deliverables

Deliverables are the artifacts, which the team agrees to develop to a specified date and submit to customer for review and approval. Deliverables on delivery date must be already seen and negotiated with customer. If deliverables upon submit date are not acceptable to customer, the issue are negotiated: why deliverables are not ready, does the team need more time, more resources, or is the scope of the deliverable not-realistic, too-big, or else.

Deliverable name	Delivery Date	Evaluation
Supplementary Specification	14 08 2005	
Use Case Specification	14 09 2005	
User Interface Specification	14 09 2005	
Initial Prototype	14 09 2005	
Test Plan	14 09 2005	
Software Architecture Document	21 09 2005	
Functional Prototype	28 09 2005	
Operational system	12 10 2005	
Deployment tools	19 10 2005	

Table G.1: Deliverables

G.1.1 Supplementary Specification

The goal of this document is to specify requirements, that are not captured directly by use cases, such as usability, reliability, performance and supportability. Also operating systems and environments, compatibility requirements, and design constraints.

G.1.2 Use Case Specification

This document captures all use-cases identified by the team and detalises them. Any identified business rules will be captured here as well.

G.1.3 User Interface Specification

The document will capture and stabilize user interfaces. The document will also include a diagram of the overall system from user perspective.

G.1.4 Software Architecture Document

Software Architecture Document will outline the software architecture to be used in the system. It will also prioritize use cases in order of implementation.

G.1.5 Test Plan

Test plan will outline the areas going to be tested. What techniques will be used to perform testing. Testing prioritization and scheduling will be defined as well.

G.1.6 Initial Prototype

This prototype will prove that a suggested architectural solution will work and solve the problem. The prototype will also solve or diminish any severe development risks identified.

G.1.7 Functional Prototype

Functional prototype will have implemented 80% of identified use-cases.

G.1.8 Operational System

System will operate 100% planned functionality. System will be tested and work without any identified erros.

G.1.9 Deployment Tools

Deployment tools will allow the deployment of operational system into customers environment. The tools must be well-tested and guarantee proper install procedure.

G.2 Human Resources

G.2.1 Team Structure

The team consist of four members. Table G.2 shows each team members details.

Name	Phone Number	E-mail address
Vilma Rudžionytė	+37060068449	vilma.rudzionyte@gmail.com
Martynas Kriaučiūnas	+37067002732	martynask@gmail.com
Viktoras Čiumanovas	+37067485615	viktoras.ciumanovas@gmail.com
Darius Damalakas	+37060246428	darius.damalakas@gmail.com

Table G.2: Team Members

G.2.2 Team Capabilities

Team good and bad traits are identified in table G.3.

G.2.3 Roles and Responsibilities

The team has divided amongself the main roles and agreed on responsibilities. See table G.4.

Name	Good Traits	Bad Traits
Vilma Rudžionytė	Does not have any sense of humour, total lack of responsibility	Does not program, changes often study programs
Martynas Kriaučiūnas	Wants to do more than he is able to, Thinks he is the best in designing	Took to much jobs in one time
Viktoras Čiumanovas	Once he knew he was mistaken, but now he realised he was wrong	No self-responsibility, no personal inspiration
Darius Damalakas	Works a lot, help to everyone	Can't work steady for a long time, not always able to accept other people with their good and bad sides, underestimates own limits and abilities.

Table G.3: Team Capabilities

G.3 Project Planning

G.3.1 Phase Plan

The project is going to be developed through 17 08 2005 to 31 10 2005. This time is split into four phases.

G.3.2 Iteration Plans

1. Iteration 1 (Inception phase) on page 88
2. Iteration 2 (Elaboration phase) on page 91
3. Iteration 3 (Elaboration phase) on page 95
4. Iteration 4 (Elaboration phase) on page 99
5. Iteration 5 (Construction phase) on page 102

Name	Main Role	Description
Vilma Rudžionytė	User Interface Designer, Tester	Designs user interfaces
Martynas Kriaučiūnas	Software Architect, Documentation Manager	Responsible for setting and assessing architectural analysis, defining reuse strategy and leading Design and Implementation Work flow
Viktoras Čiumanovas	Test Designer, Tester	Responsible for designing, implementing and performing automated tests
Darius Damalakas	Project Manager, Process Engineer	Responsible for guiding the project and ensuring everybody agrees on the working methods

Table G.4: Team Responsibilities

Date	Phase Name	Phase goals
18 08 2005 – 22 08 2005	Inception	Establish project and agree on development case
23 08 2005 – 28 09 2005	Elaboration	collect requirements, define user interface, define architectural design, implement an architectural prototype
29 09 2005 – 31 10 2005	Construction	Build an operation system, test system

Table G.5: Phase Plan

G.3.3 Project Plan

Project plan includes not only deliverables, but also artifacts, which are the documents or code entities which the team will produce during the development process. Artifacts are to be used only inside the team and will not be submitted to the customer. Artifacts help the team to carry out the project by capturing various information such planning and risk management.

Some artifacts put together will constitute a deliverable, which will be shown in bold. For the deliverable dates see deliverable only list G.1.

The dates for artifacts can be changed in team freely, when ether the whole team agrees to do so in accordance with deliverable plan. The dates for deliverables can not be changed, unless through formal change request procedure.

G.4 Technical Process Plans and Guidelines

G.4.1 Iteration Management

Iteration Management plan can be found on page 106

Artifact or deliverable name	**References**	Delivery Date
Development Case	"Development Case.ppt"	20 08 2005
Software Development Plan	(this document)	22 08 2005
Iteration Management Plan	F	22 08 2005
Risk Management Strategy	chapter H	26 08 2005
Project Management Risk List	chapter I	26 08 2005
Development Risk List	chapter K	26 08 2005
Supplementary Specification	chapter M	**Deliverable**
Database Package	code	09 09 2005
Permission package Project Builder Core package	code	12 09 2005
Software Architecture Document	chapter L	13 09 2005
Database Scripts	code	13 09 2005
Test Plan	Chapter Q	**Deliverable**
User Interface Specification	chapter **??**	**Deliverable**
Use Case Specification	chapter N	**Deliverable**
Project Detail Builder	code	13 09 2005
Initial Prototype	code	**Deliverable**
Task Builder Task Detail Builder	code	18 09 2005
Event Builder Event Detail Builder	code	24 09 2005
Functional Prototype	code	**Deliverable**
Operational System	code	**Deliverable**
Deployment Tools	tools	**Deliverable**

Table G.6: Artifacts

Appendix H

Risk Management Strategy

H.1 Introduction

The purpose of this document is to explain how all the risk will be managed during the project.

H.2 Document Holder

Martynas is responsible for managing this document.

H.3 What is a Risk

A Risk is the potential harm that may arise from some present process or from some future event.

There are different kinds of risks:

- *indirect* – risks which may or may not occur and we cannot prevent them, e.g. the world may end tomorrow, it also may not end. We will not try to identify risks like this in our project and will deal with them only when such a risk occurs.
- *direct risks* – risks which may occur, can be foreseen and a mitigation strategy can be developed. So the purpose of *Risk Management Strategy* is to specify how project's risks must be managed.

H.4 Specifying a Risk

During the project we will have two huge main activities: *Project Management* and *Development* along with their own risks. Despite the similar management of these risks, they will be separated and specified in different risk lists for the clearness and simplicity reasons:

- A list of risks applied for the project management activity can be found on page 119.
- A list of risks applied for the development phase can be found on page 123.

When a direct risk is identified it must be specified instantly so it would not be forgotten. For defining risks a common template will be used:

Name: A short name for a risk (may be replaced with a document's subsection name).

Description: A more decent description of a risk.

Severity: A measure of a damage which could be caused for the project if the risk occurs. Possible values: *low*, *normal*, *high*.

Probability: A measure of chance for this risk to occur. Possible values: *low*, *normal*, *high*.

Status: Indication of a status of the risk: *active*, *solved*.

Manager: A team member who is responsible for solving the risk.

Mitigation strategy: Description of mitigation strategies for the risk.

H.5 Risk Management Strategy

As it was stated above risks can cause serious troubles for the project, so they must be solved or a mitigation strategy must be developed for them as soon as possible.

Risks should be managed according to their severity: more dangerous should be examined first. Therefore risk lists should be reviewed before every iteration

and a time must be scheduled for solving most important ones. For example almost all development risks should be solved before or in *Analysis and Design* phase so they would not influence a system architecture.

Appendix I

Project Management Risk List

I.1 Introduction

The purpose of this document is to list risks related to Project Management. Risks are managed according to the rules stated in chapter H.

I.2 Document Holder

Darius Damalakas

I.3 Risk List

I.3.1 Mis-conceived or Undefined Project Scope

Description: The team has agreed to port the existing system to .NET platform until 1st of October. The problems is how to evaluate, when the project is successfully ported. What functionality must be present and which not? Which functions are considered as "add–on" functions and will be tackled in Construction phase II (see project phase plan G.3.1). What about bug penetration? Must the port be completely bug-free? Is this feasible at all?

Severity: *very high.*

Probability: *high.*

Status: *active.*

Manager: Darius Damalakas .

Mitigation strategy: For tackling this risk the following procedure and methods will be used throughout the project.

- Agree on essential and non-essential functionality. Define how much of essential and non-essential functionality must be implemented before specific date. State this in Software Development Plan under section Project Scope G.1. Agree on Testing and bug penetration level and scope.
- Meetings with customer must be held as specified below:
  1. After meeting, create a document with captured information.
  2. In team, agree on the contents of the document.
  3. Submit the document to customer for review and approval.
  4. After the document is approved, the team must include in document what actions will be taken for the discussed items.
  5. Submit the document to customer for review and approval.

I.3.2 Low-risk Tasks go First

Description: The problem occurs, when the team spend time on solving low-risk tasks first, and leave the high tasks for the end of the project. Even if 50% of low risk tasks are solved after spending 50% of time assigned for the project, the high risk tasks might "drown" the project.

Severity: *very high.*

Probability: *high.*

Status: *active.*

Manager: Darius Damalakas .

Mitigation strategy: During iteration planning procedures, the team must choose highest risk jobs first. As an example consider whether to do a high-risk prototype before the necessary documents are ready. Good documentation indeed might help drive prototype building activities, though consider that building a prototype the team will from time to time stumble on severe technical problems, when no real advancement programming can be done. This time could be spent on making the documents ready.

Appendix J

Glossary

J.1 Document Holder

Viktoras is responsible for this document.

J.2 Glossary

1. System – "Project Register" system.
2. Employee – The company's worker, whom has right to sign-in into the system.
3. Project Manager – a user who owns a project. He is responsible for approving tasks and can edit all the information related to the project.
4. Department Manager – a user who is in charge of employees in his/her department. He/she approves tasks for his/her employees.
5. Task – A smaller activity of a project, which must be performed during its (project) running period to complete the project properly.
6. Event – The noteworthy file (e.g. system agreement) or occurrence during the project.
7. Company's resources– The company's workers work hours, which can be used for company's needs.

Appendix K

Development Risk List

K.1 Introduction

The purpose of this document is to list risks related to analysis and design, and implementation activities. Risks are managed according to the rules stated in chapter H.

K.2 Document Holder

Martynas is responsible for managing this document.

K.3 Risk List

K.3.1 Lack of .Net Knowledge

Description: None of our team members has ever developed a project using *.Net*, only Vilma has had a course about *ASP.Net*.

Severity: *very high.*

Probability: *high.*

Status: *being solved.*

Manager: Martynas.

Mitigation strategy: If not solved this risk can be a reason for a failure, therefore few mitigation strategies will be applied:

- Intense learning. We started looking for material about *.Net* before the project, have found some useful books and not are reading.
- Despite the fact all the members are studying *ASP.Net*, Martynas will put even more effort on investigating this field so he could consult the rest of the team in more difficult situations.
- External reviews. We are planing to ask few external people to review our job related to *.Net*: design documents and source code.

K.3.2 Inappropriate Database Design

Description: We had never done before a database design for a real world system. This DB design might exhibit needs with which we had never had any encounter before.

Severity: *Normal.*

Probability: *high.*

Status: *Solved.*

Manager: Martynas.

Mitigation strategy:

- Focus on this problem first and ask guidance from external sources.

Appendix L

Software Architecture

L.1 Introduction

The goal of this document is to identify and describe main parts of the system (including database and web application). This document is being written (and later constantly updated) at the end of the *Elaboration* phase so it would provide a clear vision of the system for developers.

L.2 Document Holder

Martynas Kriaučiūnas

L.3 Architectural Goals

The main purpose of making systems's architecture is to solve biggest architectural risks and find out how to implement main requirements.

Requirements for the system can be extracted from *Supplementary Specification*M and *Use Case Specification*N.

The most important requirements are:

- Data versioning. System must track all changes and provide change log when needed;

- User permissions. System must limit user's possible actions according to it's rights (check *Use Case Specification*N for more details).
- Usability. It should be possible to change user interface according to users' feed back collected during testing period.

L.3.1 Main Components

Two main components which are identified in the system:

- Web Application. This part will provide interface for end users and will control the logic (e.g. permissions).
- Data Base. All internal data will be stored in a data base.

L.3.2 Database

The main requirement which influences a design of the database is the wish to keep track of the changes.

Three different database design strategies were discussed (see *Project Report* document for more details). After considering advantages and drawbacks of all the strategies, it was chosen to duplicate a record in an other table even a single field is altered.

Detailed Description of the Design Strategy

Main rules how to add logging to a database:

1. For every table A a table AH must be created containing all the columns from table A and an extra one – $Parent$ – external key to table A;
2. If a table A has an external key to table B, then this key in table AH show to table BH.
3. When a table A is altered, new data is written to AH as well. If an external key from table A to table B exists, when the external key from table AH must show to the top child record of the table BH;

Example. For the simplicity reasons lets consider a database containing only two related tables (see figure L.1).

Figure L.1: "Projects & Employees" Database

In this example table P (P could stand for $Projects$) has an external key to table E (E could stand for $Employees$).

In order to add history control, this data base should be extended in such a way (see figure L.2).

Figure L.2: Versioned "Projects & Employees" Database

Retrieving Historical Data

If you need to get the change log for a record in table A which has $ID{=}id$, you need to look for all the rows in table AH where $Parent{=}id$. An example of SQL query:

```
SELECT * FROM AH WHERE Parent=@id;
```

If a table A has an external key (e.g. field $ExtKey$) to table B, then AH has an external key to BH. In order to get a child row you have to perform such steps:

- Find an id of the next version of the same record. This record holds the key to the needed child row;
- Retrieve a child row.

L.3.3 Web Application

This is the main and very complex part of the system.

The designing is usually started by distinguishing related parts of the system which later usually become into packages. When designing it is essential to know limitations ant capabilities of the developing tools (e.g. programming language) (it was a problem for us) which will be used. After long *trial and error* process we decomposited the system according to two different aspects:

- Data View. Divided system into parts according to the data they deal with, e.g. "Projects", "Tasks" etc.
- Interface View. Divided system into parts according to user interface, e.g. "BatchView", "Insert/Edit/View" etc.

Packages

Web application is divided into packages with subpackages:

- User Interface. This package holds all the logics (e.g. navigation between windows) and user interface part.
- Data package. This package has a two subpackages: DBFillers and Controllers. They both constitute a middle layer between *Data Base* and *User Interface*.
- User. This package is repsonisble for permission control.
- PRControls. This package is an extension to .NET user interface controls

User Interface

According to requirements, we tried to make a customizable interface and, if possible, extract any code which might be reused. It is a very good practice to avoid *copy & paste* by extracting as much functionality as possible to abstract classes. So, if system design is done well, you need to change something only in one place if a bug arises.

All the pages must be derived from the *BasePage*. This class is responsible just for few base functions, such as controlling the life cycle of a web page.

Figure L.3: System Architecture

For every group of similar pages a base class is made which holds only main logic (without any user interface design) for particular group of pages:

- *BatchView*. This class is used when a page which shows a table with some kind of data is created (e.g. to show list of project).It is responsible to fill in data to a page, redirect to respective *Edit* page and so on.
- *Edit*. A page, derived from this class, can be either *Edit* or *Insert* page. The difference is that, if you are editing, the page is filled up with a data on load.
- *View*. This group of pages allows to view a particular data in detail. The *Edit* group can be used for this reason as well, but they are separated for GUI customization reasons.

At first it was an idea to include some user interface design into base classes (there are some parts which are similar for groups of pages, e.g. page header), but, since ASP.NET does not allow to inherit ASP pages, we decided to make all the user interface design in bottom classes. So every page in the system is made separately and only *code behind* is derived from a respective class.

Data Package

Data Package contains two subpackages: DBFillers and Controllers.

DBFillers is responsible for:

- Retrieving required data for *User Interface* package;

There is an abstract base class *DBFiller*, which defines an interface. Each subclass of a *DBFiller* will fill a dataset with some data from the table.

Controllers

This package controls data submission to databse actions, ensuring History requirement is satisfied and hiding history implementation features from programmer.

There is an abstract base class *AbstractController* which defines an interface and performs most common tasks. All base classes in the *User Interface* package work with data using an instance of this class.

PRControls

This package provides new or enhanced web controls for user interface design. And example of these would be FilterDropDownList or FilterTextBox.

These controls are used only by User Interface package.

Appendix M

Supplementary Specification

M.1 Introduction

The goal of this documents is to capture requirements, which are not captured in use-cases, such as usability, reliability, performance and supportability. Also operating systems and environments, compatibility requirements, and design constraints.

M.2 Document Holder

Martynas is responsible for managing this document.

M.3 Requirements

M.3.1 Development Environment

The project must be developed using *ASP.Net* and *C#*.

M.3.2 Persistency

System will have to use Microsoft SQL Enterprise Server 2003 for storing data.

M.3.3 User Interface

The system must be viewable using Microsoft Internet Explorer.

Project's company stressed usability requirements which will be ensured doing user interface prototypes and constantly giving users to do reviews.

M.3.4 Security

System will have to use role-based security model and limit it's functionality according to a user's rights. A user can be assigned to several groups at once and posses all the rights applicable for all the groups he/she belongs to. The system must support these user types:

User – An ordinary system's user. Specific rights/limitations are defined in Use Case Specification (chapter N); An ordinary user can perform any role in the system: employee, project manager or department manager. See glossary J for details;

Administrator – can create/delete/administrate system users. Can perform any action in the system.

Windows integrated authentication must be used to identify a user.

M.3.5 Accurateness

Projects' and tasks' start and end time has to be specified using full dates: year, month and day.

Scheduled and real time which has been spent on doing a job has to be specified hourly.

M.4 Functional Requirements

M.4.1 Data Versioning / History

Problem and

For the project management purposes it is useful to have information about changes which have been made during the project and their reasons. E.g. why a task's duration time was extended and so on.

Requirement

Build an architecture which supports History, but do not build any interface functionality to actually use this feature. This feature will be brought into use after our first release.

Constraitns

The use of History feature will be developed during the second release building.

M.4.2 Multi-attachment Ability

Requirement

It is useful to attach more than one file. For example Project might have attachments to as many documents as needed. Maintenance (Prieziuros Ivykiai) could also have many attachments.

Constraints

Not applicable in first release.

M.4.3 Filtering Boxes in Datagrid

Requirement

This requirement came after discussing systems User Interface. Customers suggested that would be very great to have filtering boxes directly in header section of data grid.

Constraints

Must be developed in first release.

Appendix N

Use Case Specification

N.1 Introduction

This document contains domain model diagram, and any business logic identified, and use case specifications.

N.2 Document Holder

Viktoras is the holder of this document.

N.3 Domain Model Diagram

The domain model captures the real world classes in the problem domain. Domain model visualize the most important relations between classes together identifying the multiplicity and attributes of classes. The diagram is shown in figure N.1.

N.4 Business Logic

N.4.1 Task Creation Cycle

Task creation cycle emphasizes that the task must be approved by different users in different cases. There are three cases. One is when a task is inserted

Figure N.1: Domain model

by an employee and thus the task must be approved both by project manager (PM) and department manager(DM). The other two cases are when either the PM creates task and DM approves it, or DM creates task and PM approves it. This is summarized visually in figure N.2

N.5 Use Case Template

Access Rights: List of actors, who can initiate this use case, e.g. *Project Manager*, *Department Manager*, *Employee*, *System Administrator*.

Preconditions: Condition, which needs to start use case (e.g. a system user is authenticated and has enough rights to complete the task).

Postcondition: Condition, by which is possible to recognize the use case is performed.

Basic flow: Describes main work sequence during the use case, e.g.:

1. Project Manager inserts new project;

Figure N.2: Create Task Cycle

2. Project Manager inserts new tasks;
3. etc.

N.6 UC1: Insert Project

Postcondition: A new project is created.

Basic flow:

1. User fills in necessary information (look at table *Project* in diagram N.1);
2. User submits data;
   (a) If data is invalid, system displays message and allows to correct changes;
3. System creates a new project.

N.7 UC2: Log Into System

Preconditions: User is not logged into the system

Postcondition: User is logged into system and system does not ask to log in again for as long as user is using system, and for 30 min after user is not using the system.

Basic flow:

1. User is automatically authenticated by Windows Login.

Alternative flow:

In case the browser does not support Windows automatic authentication:

1. User enters user name and password;
2. User submits data;
3. Systems logs user into system.

N.8 UC3: Insert Task

Preconditions: User is logged into the system.

Postcondition: A new task is created.

Basic flow:

1. User fills in necessary details (look at table *Task* in diagram N.1);
2. User submits data;
   (a) If data is invalid, system displays message and allows to correct errors.
3. System creates a new task;
4. System sends e-mails to the Employee, Project Manager ant Department Manager notifying about the new task (if the task was not entered by himself).

N.9 UC4: View Projects

Preconditions: User is logged into the system.

Postcondition: Based on the filter, system displays project information.

Basic flow: Based on the employee's personal filter, system displays such a data for each project:

1. Project code;
2. Project name;
3. Project leader;
4. Customer;
5. Project Start date;
6. Project End date;
7. Is project closed (yes/no).

N.10 UC5: View Project In Detail

Preconditions: User is logged into the system and a project is selected.

Postcondition: System displays project information.

Basic flow: The system displays all the data about a project (look at table *Project* in diagram N.1).

N.11 UC6: View Events

Preconditions: User is logged into the system.

Postcondition: Based on the filter, the system displays event information.

Basic flow: The system display such data for a event:

- Name;
- Type;
- Date;
- A report applied for the Event.

N.12 UC7: View Event in Detail

Preconditions: User is logged into the system and an event is selected.

Postcondition: Based on the filter, the system displays event's information.

Basic flow: System displays all the data about the event (look at table *Event* in diagram N.1).

N.13 UC8: View Task in Detail

Preconditions: User is logged into the system and a task is selected.

Postcondition: Based on the filter, the system displays task's information.

Basic flow: System display all the data about the task (look at table *Task* in diagram N.1).

N.14 UC9: Edit Task

Access Rights: Owner, Project Manager, Department Manager.

Preconditions: A task is selected for editing and is not being edited by someone else.

Postcondition: An updated task is saved and requests for approvals are sent to Project Manager and Department Manager.

Basic flow:

1. The user changes any information about the task (look at table *Task* in diagram N.1).
   (a) If data is invalid, system displays message and allows to correct errors.

Alternative flow:

AF1 User can cancel editing at any time before it is successfully submitted to the system. In that case editing is terminated and the old version of the task is kept back.

N.15 UC10: View Tasks

Preconditions: User is logged into the system.

Postcondition: Based on the filter, the system displays information about task.

Basic flow: The system display such data for a task:

- Unique project code;
- Performer of the task;
- Name of the task;
- Scheduled hours;
- Scheduled beginning date;
- Scheduled end date;
- Real end date;
- Approved by Project Manager (yes/no);
- Approved by Department Manager (yes/no);
- Performing date;
- Spend hours for task performance.

N.16 UC11: View My Approvals

Preconditions: User is logged into the system.

Basic flow:

System show approvals filtered by "Approved By" = current user.

N.17 UC12: Approve Task

Access Rights: Project Manager, Department Manager.

Preconditions: The top version of the task is not approved by the user.

Postcondition: The task is marked as approved by the user.

Basic flow:

1. The user approves the task.

N.18 UC13: Delete Task

Access Rights: Owner, Project Manager.

Preconditions: A task is selected.

Postcondition: The task is permanently deleted from the system.

Basic flow:

1. The user asks for deleting a task and approves it (extra precautions are needed!);
2. System deletes the task and all the related data: its approvals, files etc with respective history.

N.19 UC14: Delete Project

Access Rights: Project Manager, Department Manager.

Preconditions: A project is selected.

Postcondition: The project is permanently deleted from the system.

Basic flow:

1. The user asks for deleting a project and approves it (extra precautions are needed!);
2. System deletes the project and all the related data: its tasks, approvals, files etc. with respective history.

N.20 UC15: Edit Project

Access Rights: Project Manager, Department Manager.

Preconditions: User is logged into the system.

Postcondition: The project data is edited and updated.

Basic flow:

1. The user changes any information about the project (look at table *Project* in diagram N.1).
   (a) If data is invalid, system displays message and allows to correct errors.

Alternative flow:

AF1 User can cancel editing at any time before it is successfully submitted to the system. In that case editing is terminated and the old version of the project is kept back.

N.21 UC16: Delete Event

Access Rights: Project Manager, Department Manager.

Preconditions: An Event is selected.

Postcondition: The Event is permanently deleted from the system.

Basic flow:

1. The user asks for deleting a event and approves it (extra precautions are needed!);
2. System deletes the event with its history.

N.22 UC17: Edit Event

Access Rights: Project Manager, Department Manager.

Preconditions: An Event is selected.

Postcondition: The project data is edited and updated.

Basic flow:

1. The user changes any information about the event (look at table *Event* in diagram N.1).
   (a) If data is invalid, system displays message and allows to correct errors.

Alternative flow:

AF1 User can cancel editing at any time before it is successfully submitted to the system. In that case editing is terminated and the old version of the project is kept back.

N.23 UC19: Filter

Preconditions: A Task, Project, Event or Approvals view is selected;

Postcondition: A required part of the view is shown;

Basic flow:

1. The system provides a possibility to filter the view according to all its fields and external keys;
2. The user specifies filtering criteria he/she is interested in.
3. The user submits the filter.

Appendix O

Company, School and Team Contract

Company Blue Bridge Jasinskio 16A LT – 01112, Vilnius

Contact person: Dalius Masalas

Tel: (8 5) 252 6060

E-mail: dalius.masalas@bluebridge.lt

Group members:

Darius Damalakas – darius.damalakas@gmail.com

Martynas Kriauciunas – martynas.k@gmail.com

Viktoras Ciumanovas – viktoras.ciumanovas@gmail.com

Vilma Rudzionyte – vilma.rudzionyte@gmail.com

Project period August – November 2005

Title "Project Registry"

Agreement

The group, Blue Bridge and Roskilde Business Academy are establishing this agreement as a means of protecting all parties. This agreement will remain fixed between all parties throughout the whole project period

The following items are drawn up to outline the responsibilities of all parties involved in the project:

1. The group primary focus will be on delivering a study project (written documents) for the purpose of the final fifth semester examination at

Roskilde Business Academy. Therefore the completion of any software products resulting from the project cannot be guaranteed.

2. Blue Bridge owns the commercial rights of the delivered (code and report).

3. Should Blue Bridge decide to no longer support the project: " Group would have permission to continue the project using all information already provided by Blue Bridge. " Furthermore all the references to the company can be used by the group until the end of the project.

4. Should group decided to no longer support the project: " Blue Bridge will retain all the company material, if any, up to point. " Blue Bridge owns the commercial rights of finished or unfinished code produced up to that point. " Any commercial use of unfinished product will require written consent from Blue Bridge.

5. The group grants permission to Roskilde Business Academy to publish the report at the school library for educational purposes. Should any student or teacher at the school choose to use any part of the code and/or report for any purpose aside from educational purposes, express written consent from all group members and Blue Bridge is required.

By signing this contract, I hereby state agreement and future compliance with all items contained herein.

For and behalf of Blue Bridge:

- `Dalius Masalas`

For and behalf of Roskilde Business Academy:

- `Michael Claudius`

Group members:

- `Vilma Rudžionytė`
- `Darius Damalakas`
- `Viktoras Čiumanovas`
- `Martynas Kriaučiūnas`

Appendix P

Development Case

Projektų Registras

Development case

Darius Damalakas 16 08 05

Click on an area of the screen for more information.

Week 33 - 34 (08 18 –08 22)

Week 34 – 36 (08 23 – 09 10)

Construction I
Week 37-40
(09 12 – 10 07)

Construction II
Week 41 – 53
(10 10 – 10 28)

Requirements

Artefacts (deliverables)

Supplementary Spec.
Use Case Model
Stakeholder Requests
Risk List
Domain Model

Activities

Develop Vision
Capture Common Vocabulary
Identify Stakeholders
Capture Stakeholders Requests
Identify Use Cases
Prioritize Use Cases
Detail Use Cases

Analysis & Design

Artefacts (deliverables)	Activities
Software Architecture Document •Use Case View •Data View •Architectural Goals •Logical View Prototype	Use Case Analysis Architectural Analysis Construct Prototype Design Components Design Database

Implementation

Artefacts (deliverables)	Activities
Integration Build Plan	Plan System Integration Implement Component Perform Unit Test Review Code Integrate Subsystem

Test

Artefacts (deliverables)

Test Plan
Test Case
- Test Procedures
- Test Scripts

Activities

Plan Test
Design Test
Implement Test
Execute Test Cases
Evaluate Test

Project Management

Artefacts (deliverables)

Iteration Plan
Software Development Plan
- Project Deliverables + dates
- Project Organization
- Phase Objectives + dates
- Programming Guidelines
- Product Acceptance Plan

Report Status
Development Case

Activities

Assign Project Review Authority (PRA)
Approve Project Acceptance
Develop Software Development Plan (SDP)
Plan Next Iteration
Approve Milestones
Assess Iteration
Phase Assessment (Close-out Phase)
Report Status To PRA

Roles

Analyst
•System Analyst
•Requirements Specifier (details use cases)
•User Interface Designer

Developer
•Software Architect
•Database Designer
•Designer
•Implementer
•Integrator

Tester:
•Test Designer
•Tester

Manager:
•Process Engineer
•Project Manager
•Project Reviewer

Other:
- Stakeholder
- System Administrator
- Tool Specialist

Inception phase

Milestone: Lifecycle Objectives

Evaluation Criteria
1. Agreement that the right set of requirements have been captured and that there is a shared understanding of these requirements.
2. Agreement that the cost/schedule estimates, priorities, risks, and development process are appropriate.
3. All risks have been identified and a mitigation strategy exists for each.

Objectives:
1. Establishing the project's software scope and boundary conditions, including an operational vision, acceptance criteria and what is intended to be in the product and what is not.
2. Discriminating the critical use cases of the system, the primary scenarios of operation that will drive the major design trade-offs.
3. Estimating potential risks (the sources of unpredictability)
4. Preparing the supporting environment for the project.

Inception phase

Week 33 - 34
(08 18 –08 22)

Essential Artefacts:

Risk List
Vision
Software Development Plan
Tools
Use Case Model
Supplemenetary Specification

Elaboration phase

Milestone: **Lifecycle Architecture**

Evaluation Criteria:

1. The product Vision and requirements are stable.
2. The architecture is stable.
3. Executable prototypes have demonstrated that the major risk elements have been addressed and have been credibly resolved.
4. All stakeholders agree that the current vision can be met if the current plan is executed to develop the complete system, in the context of the current architecture.

Objectives:

1. To ensure that the architecture, requirements and plans are stable enough, and the risks sufficiently mitigated
2. To establish a baselined architecture derived from addressing the architecturally significant scenarios, which typically expose the top technical risks of the project.
3. To produce an evolutionary prototype of production-quality components, as well as possibly one or more exploratory, throw-away prototypes

Essential Artefacts:

Prototypes
Risk List
Project-Specific Templates
Tools
Software Architecture Document
Design Model
Data Model

Features	Benefit	Effort	Risk	Architecture Impact	Stability
FEATURE1: Save and restore sort and filter criteria	Med High	Low	Low	Low	High
FEATURE2: Ability to save a RequisitePro document as a Microsoft® Word® document.	Med High	Low	Low	Low	High
FEATURE3: Ability to see deleted requirements in a view window.	Medium	Med High	Medium	Low	Medium
FEATURE4: Support for Currency datatype attributes.	Medium	Medium	Med Low	Low	Medium
FEATURE5: Support the "All" document type (provides an easy way to define	Med High	Medium	Medium	Low	Med High

Quality Assurance?
Business Modeling?
Deployment?

The capsule designer role focuses on ensuring the system can respond to events in a timely manner, in accordance with concurrency requirements. The primary vehicle for solving these problems is the Artifact: Capsule.

Appendix Q

Test Plan

Q.1 Introduction

The test plan contains information about the purpose and goals of testing within the project. Additionally, the test plan identifies the strategies to be used to implement and execute testing and resources needed.

Q.2 Document Holder

Darius is the holder of this document.

Q.3 Types of Tests

Q.3.1 Functional Tests

Functional requirements for test, as their name implies, are derived from descriptions of the systems functional behaviors. At a minimum, each use case should constitute at least one requirement for test.

TestObjective:

> Ensure proper functionality, including navigation, data entry, processing, and retrieval.

Technique:

Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following:

- The expected results occur when valid data is used.
- The appropriate error or warning messages are displayed when invalid data is used.
- Each business rule is properly applied.

Completion Criteria:

- All planned tests have been executed.
- All identified defects have been addressed.

Q.3.2 User Interface Tests

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions. In addition, UI testing ensures that the objects within the UI function as expected and has a common style.

TestObjective: Verify the following:

- Navigation through the target-of-test properly reflects business functions and requirements, including window-to-window, field-to-field, and use of access methods (tab keys, mouse movements, accelerator keys.
- Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.

Technique:

Create or modify tests for each window to verify proper navigation and object states for each application window and objects

Completion Criteria: Each window successfully verified to remain consistent within acceptable standard.

Q.3.3 Security Testing

Security testing focus on the area of System-level Security, more specifically logging into the system.

System-level security ensures that only those users granted access to the system are capable of accessing the system.

TestObjective: Verify that users are correctly logged into system, and only those users, who have granted access.

Technique:

Create or modify tests for each window to verify proper navigation and object states for each application window and objects

Completion Criteria: Each window successfully verified to remain consistent within acceptable standard.

Appendix R

Team Contract

R.1 Purpose of this Document

The team working contract is done for several reasons:

- To support the project team by identifying human, technical and environmental resources available to the project
- To allow team members to identify and accept the realities of the project: its scope, duration, rules, interested parties and goals and their roles in the project
- To provide a simple, clear starting point for the project and a foundation for the team's working practices and social dynamics.

R.2 What is the Goal for the Team Work (What is to be Delivered to Who)

The main objective of the project is to gain experience working with C#, RUP, UML, web development and relational databases.

The goal is to complete the study project in time and to deliver Study Project documentation and System Development documentation to the school for evaluation at the final exam for this 5th semester.

The criteria for success have been included to aid us in evaluating the results of the project:

- A report outlining the decisions made and tasks performed during the completion of the research and system development project is delivered.
- The project is completed on time.
- The project receives a good evaluation from the project adviser and censor.

R.3 Team Members

Name	Telephone	E-mail
Vilma Rudzionyte	+37060068449	ragana.v@gmail.com
Viktoras Ciumanovas	+37067485615	viktoras.ciumanovas@gmail.com
Darius Damalakas	+37067348362	darius.damalakas@gmail.com
Martynas Kriauciunas	+37067002732	martynas.kriauciunas@mif.vu.lt

R.4 Resources Available for the Team and Assessment of Strengths and Weaknesses

We are the best team and here are our weaknesses and strengths

R.5 Work Organisation

This project is being done by a group therefor all the group members are equally responsible for the success and are expected to do their best.

The roles in our team were not defined artificially, i.e. the team naturally observed and noted that one member is more liable to prompt action, the others are more liable to anger, and the third one is natural idler.

Thus, the table R.2 shows team member role assignment along with ...

R.6 What is the Goal for Team Work

The main objective of the project is to gain experience working with UP, UML, C#.NET.

Name	Good Traits	Bad Traits
Vilma Rudžionytė	Does not have any sense of humour, total lack of responsibility	Does not program, changes often study programs
Martynas Kriaučiūnas	Wants to do more than he is able to, Thinks he is the best in designing	Took to much jobs in one time
Viktoras Čiumanovas	Once he knew he was mistaken, but now he realized he was wrong	No self-responsibility, no personal inspiration
Darius Damalakas	Works a lot, help to everyone	Can't work steady for a long time, not always able to accept other people with their good and bad sides, underestimates own limits and abilities.

Table R.1: Team Capabilities

The goal is to complete the study project in time and to deliver Study Project documentation and Systems Development documentation to the school for evaluation at the final exam for this semester.

The criteria for success have been included to aid us in evaluating the results of the project:

- A report outlining the decisions made and tasks performed during the completion of the research and system development project is delivered;
- The project is completed on time;
- The project receives a good evaluation from the project advisers and censor.

Name	Main Role
Vilma Rudžionytė	User Interface Evaluation, Project Report Designs and evaluates UI
Description	She has done UI design in 3rd semester. She easily copes with small-scope tasks, but easily bumps into dead-ends when given more abstract tasks. Needs strong guidance
Martynas Kriaučiūnas	**Software Architect:** Responsible for setting and assessing architectural analysis, defining reuse strategy and leading Design and Implementation Work flow
Description	He has outstanding experience in designing software and building reusable architectures. 4 years of experience with real world projects. Critical thinking (some times too much). Structural thinking
Viktoras Čiumanovas	**User Interface Design:** Responsible for building UI prototype and communicating with customers
Description	He has outstanding experience in designing software and building reusable architectures. 4 years of experience with real world projects. Critical thinking (some times too much). Structural thinking
Darius Damalakas	Project Manager, Process Engineer Responsible for guiding the project and ensuring everybody agrees on the working methods
Description	He has outstanding experience in designing software and building reusable architectures. 4 years of experience with real world projects. Critical thinking (some times too much). Structural thinking

Table R.2: Team Roles

R.7 Technical Resources Available

The following is a list of resources owned by group members or available from Blue Bridge (BB):

R.8 Software

- LaTeX (Word Processor)
- Dia (Diagram Application)
- Subversion (Version Control System)
- Visual Studio.NET 2003(Programming Language)
- Windows 98, Windows 2000, Windows XP home edition
- SQL Server 2000
- MS Project

R.9 Hardware

- PCs at BB and at home
- Laser printer at BB
- Internet and network connections for backup and communication

R.10 Working Place

Team members living places or in the BB.

R.11 Risks

Based on the above analysis and the experience from last semesters work typical risks in the group work fall in these categories:

- Disharmony in group work

- Sharing of workload
- Lack of skills

Appendix S

XP versus UP a Methodology Comparison Study Work

S.1 Introduction

The comparison will be made based on information from official XP web site [1] and practice gain from following the approaches described in Craig's Larman book [10] "Applying UML and patterns" and "The New Methodology" [11] from Martin Flower.

S.2 Philosophy

These sections present the philosophy the methodologies expose. In this context, "philosophy" is regarded the same as in the book "Information Systems development" [9] it is a principle or a set of principles that underlie the methodology.

S.2.1 Paradigm

Two paradigms are considered relevant to this are the science [6] and the systems [7] paradigm.

It is difficult to strictly say to what paradigm methodologies should be put. We identify UP as being science paradigm while XP being systems paradigm. This partitioning is on the biggest part influenced by the overall perception of systems development for particular methodology. XP tries to build the system gradually. On the other hand, we believe, that UP is based on science paradigm because it tries more or less to build the system which is based on elaborate architecture (because UP is architecture-centric). Of course UP (as also XP) is based on an iterative approach. The important issue is not the exact paradigm we assign, but the discussion our ideas might generate.

S.2.2 Objectives

This section distinguishes the objectives of the methodologies, or more strictly – it tries to answer the question 'could the use of methodology lead to the implementation of a purely organizational or non-IT solution?'

The objective of the XP methodology is not to come up with the non-IT or purely organizational solution, but quite the opposite – XP is designed to build IT-systems and do not address the analysis of the organization strategy and functions. UP is identified also as capable of producing only IT solution. This is based on the fact that both methodologies do not incorporate business analysis steps.

S.2.3 Domain

This section describes methodologies as being of the planning, organization, and strategy type or problem solving methodologies. This is related to the objectives of the methodology, but focuses on what aspects or domain the methodology seeks to address.

The primary focus is to solve a particular problem (to create a system which helps achieve some specific goals). That means the domain of interest is rather narrow and solely limits to creating an IT-solution and non-IT solution is not in

the area of interest. XP does not address the general planning, organizational, and strategy of information and systems in the organization analysis. UP is also described as problem-solving methodology for the same reasons.

S.2.4 Target

This section is concerned with the target system to be developed, or more particularly whether the methodology is aimed at particular types of application, types of domain, size of system, environment and so on.

The XP is suited, as stated in XP web site [1], for small groups of people, usually varying between 2 and 12, though larger groups of 30 have also report success. It is not possible to use XP with huge staff. XP enables to embrace change, so projects with highly unstable project requirements fits perfectly.

UP is considered to be general purpose methodology, although it is suggested they are not really helpful for simple, limited systems.

S.3 Model

This sections concerns with the model or models that the methodologies use. This can be described in terms of the type of the model, the levels of abstraction of the model and the orientation or focus of the model.

XP uses OO modeling techniques. It is stated that there is some written documentation for the software, more commonly interpreting requirements than documenting design.

In UP the basic models are also of object-orientation approach. However, the models are oriented not only for software design, but also business processes.

S.4 Techniques and Tools

This section describes techniques and tools used by the methodologies

S.4.1 UP Techniques

UP utilizes the following techniques in developing IT-solution

- Entity modeling

- Gannt charts
- OO techniques and UML (class diagram, use case diagram, interaction diagram, sequence and state chart diagrams, activity diagram)

S.4.2 UP Tools

UP does not specifically address the tools to be used together with methodology. Some individual tools that might be used with UP

- For diagramming: Microsoft Visio, MagicDraw,
- For project management: Microsoft project,
- Requirements: Microsoft office, Open Office, any XML technology supporting program,
- Application packages (supports whole process): Rational Rose.

S.4.3 XP Techniques

XP does not specify techniques to be used except that it is object oriented methodology and appropriate tools and techniques should be used. Based on this we can state that it uses.

- OO techniques and UML (class diagram, use case diagram, interaction diagram, sequence and state chart diagrams, activity diagram)
- Requirements are modeled with the use of CRC cards. CRC stands for Class, Responsibilities, and Collaboration. The cards are rarely filled up in great detail and in most cases only a class name is written at the top of the card.
- User stories [3]

S.4.4 XP Tools

XP does not specify any specific tools to use except the very important area – testing.

Phase/Methodology	XP	UP
Strategy	0	1
Feasibility	0	1
Analysis	1	3
Logical Design	1	3
Physical Design	1	3
Programming	3	1
Testing	3	2
Implementation	1	3
Evaluation	0	2
Maintenance	0	0

- Unit Test Framework

Unit test framework is a development tool to facilitate creating and managing tests. "Your unit test framework can help you formalize requirements, clarify architecture, write code, debug code, integrate code, release, optimize, and of course test."

S.5 Scope

Both methodologies are valued against the scope of software development life cycle stages they cover. This kind of comparison has major flaw – not all methodologies do follow a life cycle and follow other approaches. As is with our case, both methodologies use iterative approach. The values for evaluating stage coverage ranges from 0 (means methodology does not cover the stage at all) to 3 (covers in detail). The comparison is based on our quick investigation so the table should not be understood as concise and in most cases correct.

The comparison clearly shows that the UP covers most of the areas rather in detail while XP focuses on programming and testing. This is because XP is a lightweight methodology [2]. XP focuses on delivering the software when it is needed and embraces changes gently through short-time boxed iterations (early feedback) and simplicity.

UP starts with inception phase looking at whether the project is feasible,

and proceeds further with the project requirement gathering and analysis. XP does not have an implicit analysis phase – customer is available all the time and writes user stories [3]. User stories serve as a replacement of a big requirements document.

XP explicitly does not state logical or physical design stages – though spike solutions are created to help resolve various technical or system design problems. UP, on the contrary, has a design discipline, where design and data models are created with the Software Architecture document.

XP covers programming stage thoroughly. It has many, though simple rules and practices to guide the programmer of achieving simple code with extendable design. The coding rules are as follows: the customer is always available, code must be written to agreed standards, code the unit test first, all code is pair programmed, only one pair integrates code at a time, integrate often, use collective code ownership, leave optimization till last, no overtime working (instead change the project scope or timing). UP explicitly assumes that the OO programming language will be used (as is the methodology itself OO).

The last stage XP covers thoroughly is testing. Testing is one of the core roots of XP. It is even said "We would like to have more lines of test than we do of actual code" [5]. Both acceptance tests and unit tests are used. Unit test should be in most cases 100 percent automated. UP pays a great deal of attention to testing also, though not so "aggressively" as XP. In a testing discipline a test model is created with lots of test cases, and, presumably, automated tests.

S.6 Outputs

S.6.1 Introduction

The next sections will be investigations of what is actually produced in terms of deliverables at the end of each stage of methodology. XP Outputs

- User Stories [3]
- CRC cards
- Release plans

- Iteration plan
- Unit tests
- Acceptance tests

S.6.2 UP Outputs

The outputs for UP will be categorized by the UP disciplines. Only the major artifacts (deliverables) will be named here and many simple non major artifacts will be skipped.

- Business modeling
  - Domain Model
- Requirements
  - Use-Case model
  - Vision
  - Supplementary Specification
  - Glossary
- Design
  - Design Model (Use Case Realizations)
  - Software architecture document
  - Data Model
- Implementation
  - Implementation model
- Project Management
  - Software Development plan (includes Project plan, iteration plans, etc.)
- Testing
  - Test Model

- Environment
  - Development Case

S.7 Practice

S.7.1 Background

The background of the methodology broadly identifies its origins in terms of academic or commercial.

The UP has mainly an academic background, while XP is a commercial methodology. The new ways of developing software developed by Kent Beck were first tried in Daimler Chrysler and resulted as an XP methodology.

S.7.2 User Base

This topic describes how wide the methodology is used.

Not discussed in this document due to the lack of information

S.7.3 Participants

In this topic we discuss the participants involved, in some contexts these are referred to as "actors" or "stake-holders". The questions answered here will be "Who is supposed to use the methodology", "What roles do they perform". Skill levels are also addressed.

S.7.4 UP Participants

A specialist team of professional systems analysts and designers perform the analysis and design aspects and professional programmers design the programs and write the code. The system is implemented by the analysts. In order to learn UP, significant training and experience is needed.

S.7.5 XP Participants

Participants are "ordinary programmers", who do not need a Ph.D. to use XP. Part of the team are also managers and customers as well. Through XP being

a lightweight methodology [2], no intensive and expensive training is needed to wield the methodology.

S.8 Product

This section describes what is supplied when purchasing a methodology and at what cost.

S.8.1 UP Products

The author of this document was not been able to find a freely available documentation. Software to support the methodology (in activities such as configuration and change control, design and analysis) is advisable, though suitable open source software might be downloaded freely from web.

S.8.2 XP Products

The descriptive documentation of the methodology is available online and is presented to the public with no charge. XP does not require any additional software to use it.

S.9 Conclusion

The two compared methodologies XP and UP are quite different from their background.

XP is designed to be a lightweight [2] and agile methodology. It is intended to be not used in projects with huge staff (suggested team size varies from 2 to 12 people). XP focuses on following practices which makes software development go faster and remove practices which make it move slower – it means documentation is made only when it is needed.

UP is quite the opposite – it is designed for big projects and with huge staff. A thorough process must be followed in order use UP methodology.

Both methodologies embrace change, though there are some differences in the two: UP embraces change through the use iterations and planning. XP uses also an iterative approach though with much smaller iterations (one of the

reasons is because of the huge staff and the communication taking place there) thus enabling considerably fast feedbacks.

It is clear that the differences are primarily implied by the size of the staff. This allows XP to reduce documentation to the minimum and still keep the communication active and efficient.

Another underlying difference between these two methodologies is the way they treat system design. UP is based on an idea that as soon as the design is stabilized, coding planning and programming can take place. While this is a very good idea in civil engineering (where the cost of designing a bridge is 10 percent of the whole cost of the project), s[1] uggests that for a large project, only 15 percent of the project is code and unit test, an almost perfect reversal of the bridge building ratios. He states that the design part takes not less than 50 percent of the work. XP addresses this (and all agile methodologies [8]) by focusing on coding thus limiting design (and documentation) activities.

UP is

- architecture centric
- Use-case driven
- Iterative
- Deals high risk issues in first place
- Object oriented

XP emphasizes:

- Team work
- Customer satisfaction by delivering software which is needed when it is needed
- Improvement of software project in four essential ways: communication, simplicity, feedback and courage
- Writing tests before actual code to be tested
- Implementing things you need right now, not the things you will need.

[1] McConnell, http://www.amazon.com/exec/obidos/ASIN/1556159005

- Do documentation whenever it is needed
- The quality of the source code
- Small iterations
- Customer involvement

Whilst there are many differences, the authors personal opinion is of that these two methodologies do not compete with each other – both stresses different parts of IS development and thus provide different approaches and practices.

S.10 References

[1] XP website

www.extremeprogramming.org

[2] Lightweight methodologies

An excerpt from XP website [1]: "A software methodology is a set of rules and practices used to create computer programs. A heavyweight methodology has many rules, practices, and documents. It requires discipline and time to follow correctly. A lightweight methodology has only a few rules and practices or ones which are easy to follow".

[3] User stories

An excerpt from XP website [1]: "User stories serve the same purpose as use cases but are not the same. They are used to create time estimates ¡. . . ¿. They are also used instead of a large requirements document. User stories are written by the customers as things that the system needs to do for them. ¡. . . ¿ They are in the format of about three sentences of text written by the customer in the customers terminology without techno-syntax".

[4] Spike solutions

An excerpt from XP website [1]: "Create spike solutions to figure out answers to tough technical or design problems¡. . . ¿ Most spikes are not good enough to keep, so expect to throw it away. The goals are reduce the risk of a technical problem or increase the reliability of a user story's estimate".

[5] The four project values

http://www.xprogramming.com/Practices/PracValues.html

[6] Science paradigm

Checkland (1981) describes science paradigm as consisting of reductionism, repeatability, and refutation: "we may reduce the complexity of the variety of the real world in experiments whose results are validated by their repeatability, and we may build knowledge by the refutation of hypothesis".

[7] Systems paradigm

Checkland (1981) argues that human activity systems are systems which do not display characteristics of breaking down a problem into smaller parts does not break the whole system. Human activity systems has quite the opposite characteristics – emergent properties (i.e. the whole is greater than the sum of the parts) and perform differently as a whole. The systems paradigm concerns itself for the whole picture, the emergent properties, and interrelationships between parts of the whole.

[8] Agile methodology

Some time ago lightweight methodologies [2] were the term to be used instead of agile, whilst now these terms are used interchangeably. Agile methods "attempt a useful compromise between no process and too much process, providing just enough process to gain a reasonable payoff", an excerpt from Martin Flower document "The new Methodology" [11]. "Agile methods are adaptive rather than predictive, Agile methods are people-oriented rather than process-oriented".

[9] "Information Systems Development"

[10] "Applying UML and patterns".

An introduction to Object-Oriented Analysis and Design and the Unified Process. Craig Larman, 2nd edition 2002

[11] The New Methodology

Martin Fowler. http://www.martinfowler.com

Appendix T

Project Registry Source

T.1 Pages

ProjectRegistry/Pages/BasePage.cs

```
using System;
using System.Data;

using System.Web.UI.WebControls;
using ProjectRegistry.DataBase;

namespace ProjectRegistry.Pages
{
    /// <summary>
    /// Base page establishes a common interface for all pag
es. It establishes user identity
    /// on page load, and additionaly, each page must:
    /// 1. provide an AbstractController, a page id
    /// 2. PRBindData() method, which performs data binding.

    /// </summary>
    /// <remarks> This class can not be made Abstract, as
Visual studio web designer will
```

```
/// complain, that aspx page can not inherit from abstract class</remarks>
public class BasePage : System.Web.UI.Page
{

    private AbstractController controller;
    protected string PageID;

    public BasePage(AbstractController controller, string id) : base ()
    {
        this.PageID = id;
        this.controller = controller;
    }

    public AbstractController Controller
    {
        get
        {
            if (this.controller == null)
                throw new System.Exception("must override");
            return this.controller;
        }
    }

    // This is only to shut up Visual Studio
    public BasePage() : base ()
    {
        this.controller = null;
    }

    /// <summary>
    /// establish user identity and bind data
    /// </summary>
    /// <param name="sender"></param>
```

```
        /// <param name="e"></param>
        protected virtual void Page_Load(object sender, System.EventArgs e)
        {
            ProjectRegistry.Users.User.getUser(this.Session)
;
            if (!Page.IsPostBack)
                PRBindData();
        }

        protected virtual void PRBindData() {throw new System.Exception("must override");}

    }
}
```

T.1.1 BatchView Pages

ProjectRegistry/Pages/BatchViews/BatchView.cs

```
using System;
using System.Data;

using System.Collections;

using System.Web.UI;
using System.Web.UI.WebControls;
using ProjectRegistry.Pages;
using ProjectRegistry.DataBase;
using ProjectRegistry.DataBase.DataSets;
using PRControls;
using PRCommonLibrary;



namespace ProjectRegistry.Pages.BatchViews
{

    /// <summary>
    /// Details an interface for BatchView pages
    /// </summary>
    public class BatchView : BasePage
    {
        // Filtering Rules
        protected PRCommonLibrary.Persistency.FilterHashTabl
e PersistentFilter;

        // Sorting Rules
        protected PRCommonLibrary.Persistency.FilterHashTabl
e PersistentSorting;
```

```
        public BatchView(AbstractController controller, stri
ng id) : base (controller, id)
        {
            // create filtering and sorting rule tables
            this.PersistentFilter =
                new PRCommonLibrary.Persistency.FilterHashTa
ble("filtering table of " + this.PageID);
            this.PersistentSorting =
                new PRCommonLibrary.Persistency.FilterHashTa
ble("sorting table of " + this.PageID);
        }

        // an empty constructor to VS designer ;)
        public BatchView() : base () {}

        protected string ReplaceBooleanTaipNe(object val)
        {
            return ReplaceBoolean(val, "taip", "ne");
        }

        protected string ReplaceBoolean(object val, string T
rue, string False)
        {
            if (val.ToString() == "True")
                return True;
            else
                return False;
        }

        override protected void OnInit(EventArgs e)
        {
            // supply session object to Filter and Soring ta
bles
```

```
            // this must be done in init stage, as the same
filter and sort rule objects
            // will persist through many sessions
            this.PersistentFilter.PageSession = this.Session
;
            this.PersistentSorting.PageSession = this.Sessio
n;
            base.OnInit(e);
        }

        /// <summary>
        /// adds any filters programmaticaly.
        /// </summary>
        /// <remarks>
        /// <p> For ex. If a page is invoked somePage.aspx?P
MID=3 <br>
        /// then it could add a filter
        /// this.PersistentFilter.injectRule("ProjectID",Fil
terOperation.Equals, PMID, "");
        /// </p>
        /// <p>subclasses must always override this method.
</p> </remarks>
        protected virtual void addFilters()
        {
            throw new System.Exception("must override");
        }

        // each subpage must have a datagrid
        protected virtual DataGrid dataGrid
        {
            get { throw new System.Exception("must override"
); }
        }
```

```
        /// <summary>
        /// each subpage must have a dataview.
        /// </summary>
        /// <remarks> DataView is usually bound to dataGrid<
/remarks>

        protected virtual DataView dataView
        {
            get { throw new System.Exception("must override"
); }
        }

        // checks if page index is in correct boundaries
        private void setCurrentPageIndex()
        {
            if (this.dataGrid.CurrentPageIndex * this.dataGr
id.PageSize >= this.dataView.Count)
            {
                int page = (this.dataView.Count / this.dataG
rid.PageSize) - 1;
                // page might equal to -1, when dataView.Cou
nt=0
                if (page < 0) page = 0;
                this.dataGrid.CurrentPageIndex = page;
            }
        }

        /// <summary>
        ///
        /// </summary>
        /*protected virtual DataSet dataSet
        {
```

```
            get { throw new System.Exception("must override"
);}
        }*/

        protected override void PRBindData()
        {
            this.dataGrid.VirtualItemCount = this.dataView.C
ount;
            this.addFilters();
            dataView.RowFilter = this.PersistentFilter.build
Filter();
            dataView.Sort = this.PersistentSorting.buildFilt
er();
            DataBind();
            this.setCurrentPageIndex();
            managePermissions();
        }

        // Permissions should me managed through this method

        protected virtual void managePermissions() {}

        // Responsible for paging
        protected virtual void Paging(object source, DataGri
dPageChangedEventArgs e)
        {
            dataGrid.CurrentPageIndex = e.NewPageIndex;
            PRBindData();
        }

        /// <summary>
        /// Handle sorting commands
        /// </summary>
```

```
        protected void Sorting(object source, System.Web.UI.
WebControls.DataGridSortCommandEventArgs e)
        {
            // if alrady exists a filter rule
            if (this.PersistentSorting.getFilterRule(e.SortE
xpression) != null)
            {
                // invert sorting from ASC to DESC and vice
versa
                PRCommonLibrary.Persistency.FilterHashTable.
FilterRule fr =
                    this.PersistentSorting.getFilterRule(e.S
ortExpression);
                if (fr.value1 == "ASC")
                    fr.value1 = "DESC";
                else
                    fr.value1 = "ASC";
            }
            else // if the rule does not contain specified s
orting
            {
                // clear filter table , so no multiple sorti
ng is not allowed.
                this.PersistentSorting.removeRules();
                // add new sorting rule
                this.PersistentSorting.injectRule(e.SortExpr
ession,
                    PRCommonLibrary.Persistency.FilterHashTa
ble.FilterRule.FilterOperation.Sorting,
                    "ASC", "");
            }
            // rebind data to reflect changes
            PRBindData();
```

```
        }



        /// <summary>
        /// restores filtering controls to show correspondin
g values
        /// </summary>
        protected void DataGrid_ItemDataBound(object sender,
 System.Web.UI.WebControls.DataGridItemEventArgs e)
        {
            // search for FilterDDL or FilterTB controls in
datagrid
            if (e.Item.ItemType != ListItemType.Header) retu
rn;
            foreach (TableCell tc in e.Item.Cells)
            {
                foreach (Control c in tc.Controls)
                {
                    if (c is PRControls.FilterDropDownList)
                    {
                        // if found DDL, make it show any cu
rrent filter rules
                        PRControls.FilterDropDownList ddl =
c as PRControls.FilterDropDownList;
                        if (this.PersistentFilter.getFilterR
ule(ddl.FilterColumnName) == null) continue;
                        string Value = this.PersistentFilter
.getFilterRule(ddl.FilterColumnName).value1;
                        System.Web.UI.WebControls.ListItem i
tem = ddl.Items.FindByValue(Value);
                        ddl.SelectedIndex = ddl.Items.IndexO
f(item);
```

```
                    }
                    if (c is PRControls.FilterTextBox)
                    {
                        // if found TB, make it show text, w
hich it had before
                        PRControls.FilterTextBox tb = c as P
RControls.FilterTextBox;
                        if (this.PersistentFilter.getFilterR
ule(tb.FilterColumnName) == null) continue;
                        tb.Text = this.PersistentFilter.getF
ilterRule(tb.FilterColumnName).value1;
                    }

                }
            }
        }



        /// <summary>
        /// Binds Events to DropDownList's in DataGrid heade
r
        /// </summary>
        protected void DataGrid_ItemCreated(object sender, S
ystem.Web.UI.WebControls.DataGridItemEventArgs e)
        {
            // search for FilterDLL or FilterTB control
            if (e.Item.ItemType == ListItemType.Header)
            {
                foreach (TableCell tc in e.Item.Cells)
                {
                    foreach (Control c in tc.Controls)
                    {
```

```
                        if (c is PRControls.FilterDropDownLi
st)
                        {
                            DropDownList ddl = c as DropDown
List;
                            ddl.AutoPostBack = true;
                            ddl.SelectedIndexChanged += new
EventHandler(ddl_SelectedIndexChanged);
                        }
                        if (c is PRControls.FilterTextBox)
                        {
                            PRControls.FilterTextBox tb = c
as PRControls.FilterTextBox;
                            tb.AutoPostBack = true;
                            tb.TextChanged += new EventHandl
er(tb_TextChanged);
                        }
                    }
                }

            }
        }

        /// <summary>
        /// when PRDropDownList selection changes, we add a
filter or remove it
        /// </summary>
        protected void ddl_SelectedIndexChanged(object sende
r, EventArgs e)
        {
            if (sender is PRControls.FilterDropDownList)
            {
                PRControls.FilterDropDownList ddl = sender a
```

```
s PRControls.FilterDropDownList;
                if ((ddl.SelectedValue != null) && (ddl.Sele
ctedValue != "-1"))
                {
                    this.PersistentFilter.injectRule(ddl.Fil
terColumnName,
                        ddl.FilterOperation, ddl.SelectedVal
ue, null);
                }
                else
                    this.PersistentFilter.removeRule(ddl.Fil
terColumnName);
            }
            this.PRBindData();
        }

        /// <summary>
        /// When PRTextBox changes text, we add or remove fi
lter
        /// </summary>
        private void tb_TextChanged(object sender, EventArgs
 e)
        {
            if (sender is PRControls.FilterTextBox)
            {
                PRControls.FilterTextBox tb = sender as PRCo
ntrols.FilterTextBox;
                tb.Text.Trim();
                if (tb.Text.ToString() == "")
                    this.PersistentFilter.removeRule(tb.Filt
erColumnName);
                else
                    this.PersistentFilter.injectRule(tb.Filt
```

```
erColumnName,
                        tb.FilterOperation, tb.Text, null);
            }
            this.PRBindData();
        }
    }
}
```

ProjectRegistry/Pages/BatchViews/BatchProjectView.aspx ProjectRegistry/Pages/BatchViews/BatchProjectView.aspx.resx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using ProjectRegistry.Users;
using ProjectRegistry.DataBase.Project;
using PRControls;
using PRCommonLibrary;



namespace ProjectRegistry.Pages.BatchViews
{
    public class BatchProjectView : BatchView
    {
        protected System.Web.UI.WebControls.DataGrid dgProje
cts;
        protected System.Web.UI.WebControls.Label Label1;
        protected ProjectRegistry.DataBase.DataSets.ProjectM
anagers.PMs dsPMs1;
        protected ProjectRegistry.DataBase.DataSets.Projects
.DSProjects dsProjectView;
        protected System.Data.DataView dvProjects;
        protected PRControls.FilterDropDownList ddlPMs;
        protected System.Web.UI.WebControls.Button btInsert;
```

```
        protected ProjectRegistry.DataBase.DataSets.TrueFals
e.DSTrueFalse dsProjectStatus;
        protected System.Web.UI.WebControls.Panel Panel1;
        protected System.Web.UI.WebControls.Button Button1;
        protected System.Web.UI.WebControls.Panel Panel2;
        protected ProjectRegistry.DataBase.DataSets.Years.DS
Years dsYears;



        public BatchProjectView() : base(new ProjectControll
er(), "BatchProjectView") {}

        protected override DataView dataView
        {
            get
            {
                return this.dvProjects;
            }
        }

        protected override DataGrid dataGrid
        {
            get
            {
                return dgProjects;
            }
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
```

```
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.dsPMs1 = new ProjectRegistry.DataBase.DataS
ets.ProjectManagers.PMs();
            this.dsProjectView = new ProjectRegistry.DataBas
e.DataSets.Projects.DSProjects();
            this.dvProjects = new System.Data.DataView();
            this.dsProjectStatus = new ProjectRegistry.DataB
ase.DataSets.TrueFalse.DSTrueFalse();
            this.dsYears = new ProjectRegistry.DataBase.Data
Sets.Years.DSYears();
            ((System.ComponentModel.ISupportInitialize)(this
.dsPMs1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectView)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dvProjects)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectStatus)).BeginInit();
```

```
            ((System.ComponentModel.ISupportInitialize)(this
.dsYears)).BeginInit();
            this.dgProjects.ItemCreated += new System.Web.UI
.WebControls.DataGridItemEventHandler(this.DataGrid_ItemCrea
ted);
            this.dgProjects.ItemCommand += new System.Web.UI
.WebControls.DataGridCommandEventHandler(this.dgProjects_Ite
mCommand);
            this.dgProjects.PageIndexChanged += new System.W
eb.UI.WebControls.DataGridPageChangedEventHandler(this.Pagin
g);
            this.dgProjects.SortCommand += new System.Web.UI
.WebControls.DataGridSortCommandEventHandler(this.Sorting);
            this.dgProjects.ItemDataBound += new System.Web.
UI.WebControls.DataGridItemEventHandler(this.DataGrid_ItemDa
taBound);
            this.btInsert.Command += new System.Web.UI.WebCo
ntrols.CommandEventHandler(this.btInsert_Command);
            //
            // dsPMs1
            //
            this.dsPMs1.DataSetName = "PMs";
            this.dsPMs1.Locale = new System.Globalization.Cu
ltureInfo("en-US");
            //
            // dsProjectView
            //
            this.dsProjectView.DataSetName = "DSProjectView"
;
            this.dsProjectView.Locale = new System.Globaliza
tion.CultureInfo("en-US");
            //
            // dvProjects
```

```
            //
            this.dvProjects.Table = this.dsProjectView.Proje
cts;
            //
            // dsProjectStatus
            //
            this.dsProjectStatus.DataSetName = "DSProjectSta
tus";
            this.dsProjectStatus.Locale = new System.Globali
zation.CultureInfo("en-US");
            //
            // dsYears
            //
            this.dsYears.DataSetName = "DSYears";
            this.dsYears.Locale = new System.Globalization.C
ultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
.dsPMs1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectView)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dvProjects)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectStatus)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsYears)).EndInit();

        }
        #endregion

        private void dgProjects_ItemCommand(object source, S
```

```
ystem.Web.UI.WebControls.DataGridCommandEventArgs e)
        {
            switch (e.CommandName)
            {
                case "View":
                    this.Response.Redirect("../DetailView/Vi
ewProject.aspx?ProjectID=" + e.CommandArgument, false);
                    break;
                case "ViewTasks":
                    this.Response.Redirect("BatchTaskView.as
px?ProjectID=" + e.CommandArgument, false);
                    break;
                case "Edit":
                    this.Response.Redirect("../Inserts/Inser
tProject.aspx?ProjectID=" + e.CommandArgument, false);
                    break;
                case "Delete":
                    ProjectRegistry.DataBase.Project.Project
Controller pc =
                        (this.Controller as ProjectRegistry.
DataBase.Project.ProjectController);
                    pc.deleteProject(e.CommandArgument.ToStr
ing());
                    this.PRBindData();
                    break;
            }
        }

        protected override void PRBindData()
        {
            new ProjectRegistry.DataBase.DataSets.ProjectMan
agers.PMsDSF(dsPMs1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Projects.P
```

```
rojectsDSF(dsProjectView).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Projects.P
rojectStatusDSF(dsProjectStatus).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Years.Year
sDSF(dsYears).fillDataSet();
            base.PRBindData();
        }

        protected override void managePermissions()
        {
            AccessControl ac =
                    new AccessControl(ProjectRegistry.Users.
User.getUser(this.Session));
            for (int i = 0; i < dataGrid.Items.Count; i++)
            {
                System.Web.UI.Control cntrl = dataGrid.Items
[i].FindControl("lbEdit");
                cntrl.Visible = ac.isProjectManager(dataGrid
.DataKeys[i].ToString());
            }
        }

        protected override void addFilters()
        {
            if (this.IsPostBack == false)
            {
                if (this.Page.Request.Params["PMID"] != null
)
                {
                    string pmid = this.Page.Request.Params["
PMID"].ToString();
                    PRCommonLibrary.Persistency.FilterHashTa
ble.FilterRule.FilterOperation fo =
```

```
                        PRCommonLibrary.Persistency.FilterHa
shTable.FilterRule.FilterOperation.Equals;
                    base.PersistentFilter.injectRule("PMID",
 fo, pmid, null);
                }
            }
        }

        private void btInsert_Command(object sender, System.
Web.UI.WebControls.CommandEventArgs e)
        {
            this.Response.Redirect("../Inserts/InsertProject
.aspx", false);
        }

    }
}
```

ProjectRegistry/Pages/BatchViews/BatchTaskView.aspx ProjectRegistry/Pages/BatchViews/BatchTaskView.aspx.resx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using ProjectRegistry.Users;
using ProjectRegistry.DataBase.Task;

namespace ProjectRegistry.Pages.BatchViews
{

    /**/
    /// <summary>
    /// Summary description for BatchTaskView.
    /// </summary>
    public class BatchTaskView : BatchView
    {
        protected ProjectRegistry.DataBase.DataSets.Tasks.DS
Tasks dsTaskView1;
        protected System.Data.DataView dwTasks;
        protected System.Web.UI.WebControls.Panel MPanel;
        protected ProjectRegistry.DataBase.DataSets.Employee
s.DSEmployees dsEmployees1;
        protected System.Web.UI.WebControls.Button btInsert;
```

```
        protected System.Web.UI.WebControls.Panel Panel2;
        protected ProjectRegistry.DataBase.DataSets.TrueFals
e.DSTrueFalse dsTaipNe;
        protected System.Web.UI.WebControls.DataGrid dgTasks
;

        public BatchTaskView() : base(new TaskController(),
"BatchTaskView") {}

        /*protected override DataSet dataSet
        {
            get
            {
                return dsTaskView1;
            }
        }*/

        protected override DataView dataView
        {
            get
            {
                return this.dwTasks;
            }
        }



        protected override DataGrid dataGrid
        {
            get
            {
                return this.dgTasks;
            }
        }
```

```
        protected override void PRBindData()
        {
            new ProjectRegistry.DataBase.DataSets.Tasks.Task
sDSF(dsTaskView1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Employees.
EmployeesDSF(dsEmployees1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.TrueFalse.
TrueFalseDSF(dsTaipNe).fillDataSet();
            base.PRBindData();
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.dsTaskView1 = new ProjectRegistry.DataBase.
DataSets.Tasks.DSTasks();
```

```
            this.dwTasks = new System.Data.DataView();
            this.dsEmployees1 = new ProjectRegistry.DataBase .DataSets.Employees.DSEmploye
            this.dsTaipNe = new ProjectRegistry.DataBase.Dat
aSets.TrueFalse.DSTrueFalse();
            ((System.ComponentModel.ISupportInitialize)(this
.dsTaskView1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dwTasks)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsTaipNe)).BeginInit();
            this.dgTasks.ItemCreated += new System.Web.UI.We
bControls.DataGridItemEventHandler(this.DataGrid_ItemCreated
);
            this.dgTasks.ItemCommand += new System.Web.UI.We
bControls.DataGridCommandEventHandler(this.dgTasks_ItemComma
nd);
            this.dgTasks.PageIndexChanged += new System.Web.
UI.WebControls.DataGridPageChangedEventHandler(this.Paging);

            this.dgTasks.SortCommand += new System.Web.UI.We
bControls.DataGridSortCommandEventHandler(this.Sorting);
            this.dgTasks.ItemDataBound += new System.Web.UI.
WebControls.DataGridItemEventHandler(this.DataGrid_ItemDataB
ound);
            this.dgTasks.SelectedIndexChanged += new System.
EventHandler(this.dgTasks_SelectedIndexChanged);
            this.btInsert.Command += new System.Web.UI.WebCo
ntrols.CommandEventHandler(this.btInsert_Command);
            //
            // dsTaskView1
            //
```

```
            this.dsTaskView1.DataSetName = "DSTaskView";
            this.dsTaskView1.Locale = new System.Globalizati
on.CultureInfo("en-US");
            //
            // dwTasks
            //
            this.dwTasks.Table = this.dsTaskView1.Tasks;
            //
            // dsEmployees1
            //
            this.dsEmployees1.DataSetName = "DSEmployees";
            this.dsEmployees1.Locale = new System.Globalizat
ion.CultureInfo("en-US");
            //
            // dsTaipNe
            //
            this.dsTaipNe.DataSetName = "DSTrueFalse";
            this.dsTaipNe.Locale = new System.Globalization.
CultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
.dsTaskView1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dwTasks)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsTaipNe)).EndInit();

        }
        #endregion
```

```
        /*protected void EditCommand(object source, System.W
eb.UI.WebControls.DataGridCommandEventArgs e)
        {
            EditCommand(e, "../Edits/EditProject.aspx");
        }*/

        private void dgTasks_ItemCommand(object source, Syst
em.Web.UI.WebControls.DataGridCommandEventArgs e)
        {
            switch (e.CommandName)
            {
                case "View":
                    /*this.Response.Redirect("ViewTask.aspx?
TaskID=" +
                        e.CommandArgument, false);*/
                    break;
                case "Edit":
                    this.Response.Redirect("../Inserts/Inser
tTask.aspx?TaskID=" +
                        e.CommandArgument, false);
                    break;
                case "Delete":
                    ProjectRegistry.DataBase.Task.TaskContro
ller tc =
                        this.Controller as ProjectRegistry.D
ataBase.Task.TaskController;
                    tc.deleteTask(e.CommandArgument.ToString
());
                    this.PRBindData();
                    break;
                default:
                    break;
            }
```

```
        }

        protected override void managePermissions()
        {
            AccessControl ac =
                new AccessControl(ProjectRegistry.Users.User
.getUser(this.Session));
            for (int i = 0; i < dataGrid.Items.Count; i++)
            {
                bool canEdit = ac.canEditTask(dataGrid.DataK
eys[i].ToString());
                // Edit button
                System.Web.UI.Control cntrl = dataGrid.Items
[i].FindControl("lbEdit");
                cntrl.Visible = canEdit;

                // Delete button
                cntrl = dataGrid.Items[i].FindControl("lbDel
ete");
                cntrl.Visible = canEdit;
            }
        }

        protected override void addFilters()
        {
            // if this is not post back
            if (this.IsPostBack == true) return;
            // then maybe we got a reqyest to filter our tab
le?
            if (this.Request.Params["ProjectID"] != null)
            {
                string pid = this.Request.Params["ProjectID"
];
```

```
                this.PersistentFilter.injectRule("ProjectID"
,
                    PRCommonLibrary.Persistency.FilterHashTa
ble.FilterRule.FilterOperation.Equals, pid , "");
            }
            if (this.Request.Params["EmployeeID"] != null)
            {
                string pmid = this.Request.Params["EmployeeI
D"];
                this.PersistentFilter.injectRule("EmployeeID
",
                    PRCommonLibrary.Persistency.FilterHashTa
ble.FilterRule.FilterOperation.Equals, pmid, "");
            }
        }

        private void btInsert_Command(object sender, System.
Web.UI.WebControls.CommandEventArgs e)
        {
            PRCommonLibrary.Persistency.FilterHashTable.Filt
erRule filterRule =
                PersistentFilter.getFilterRule("ProjectID");

            if (filterRule != null)
                this.Response.Redirect("../Inserts/InsertTas
k.aspx?ProjectID=" +
                    filterRule.value1, false);
            else
                this.Response.Redirect("../Inserts/InsertTas
k.aspx", false);
        }

        private void dgTasks_SelectedIndexChanged(object sen
```

```
der, System.EventArgs e)
        {


        }


    }
}
```

ProjectRegistry/Pages/BatchViews/BatchEmployeeView.aspx ProjectRegistry/Pages/BatchViews/BatchEmployeeView.aspx.resx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using ProjectRegistry.DataBase.Employee;

namespace ProjectRegistry.Pages.BatchViews
{
    /// <summary>
    /// Summary description for BatchEmployeeView.
    /// </summary>
    public class BatchEmployeeView : BatchView
    {
        protected ProjectRegistry.DataBase.DataSets.Employee
s.DSEmployees dsEmployeeView1;
        protected System.Data.DataView dwEmployees;
        protected System.Web.UI.WebControls.LinkButton lbIns
ert;
        protected System.Web.UI.WebControls.Panel Panel1;
        protected System.Web.UI.WebControls.Panel Panel2;
        protected ProjectRegistry.DataBase.DataSets.TrueFals
e.DSTrueFalse dsTaipNe1;
        protected ProjectRegistry.DataBase.DataSets.Departme
nts.AllDepartment dsDepartments1;
```

```
        protected System.Web.UI.WebControls.DataGrid DataGri
d1;

        public BatchEmployeeView() : base(new ProjectRegistr
y.DataBase.Employee.EmployeeController(),
            "BatchEmployeesView") {}

        protected override DataView dataView
        {
            get
            {
                return this.dwEmployees;
            }
        }

        protected override DataGrid dataGrid
        {
            get
            {
                return this.DataGrid1;
            }
        }

        protected override void addFilters() {}

        protected override void PRBindData()
        {
            // fill and bind data
            new ProjectRegistry.DataBase.DataSets.Employees.
EmployeesDSF(dsEmployeeView1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.TrueFalse.
TrueFalseDSF(dsTaipNe1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Department
```

```
s.AllDepartmentDSF(dsDepartments1).fillDataSet();
            base.PRBindData();
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.dsEmployeeView1 = new ProjectRegistry.DataB
ase.DataSets.Employees.DSEmployees();
            this.dwEmployees = new System.Data.DataView();
            this.dsTaipNe1 = new ProjectRegistry.DataBase.Da
taSets.TrueFalse.DSTrueFalse();
            this.dsDepartments1 = new ProjectRegistry.DataBa
se.DataSets.Departments.AllDepartment();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployeeView1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
```

```
.dwEmployees)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsTaipNe1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsDepartments1)).BeginInit();
            this.DataGrid1.ItemCreated += new System.Web.UI.
WebControls.DataGridItemEventHandler(this.DataGrid_ItemCreat
ed);
            this.DataGrid1.ItemCommand += new System.Web.UI.
WebControls.DataGridCommandEventHandler(this.DataGrid_ItemCo
mmand);
            this.DataGrid1.PageIndexChanged += new System.We
b.UI.WebControls.DataGridPageChangedEventHandler(this.Paging
);
            this.DataGrid1.SortCommand += new System.Web.UI.
WebControls.DataGridSortCommandEventHandler(this.Sorting);
            this.DataGrid1.ItemDataBound += new System.Web.U
I.WebControls.DataGridItemEventHandler(this.DataGrid_ItemDat
aBound);
            this.lbInsert.Click += new System.EventHandler(t
his.lbInsert_Click);
            //
            // dsEmployeeView1
            //
            this.dsEmployeeView1.DataSetName = "DSEmployeeVi
ew";
            this.dsEmployeeView1.Locale = new System.Globali
zation.CultureInfo("en-US");
            //
            // dwEmployees
            //
            this.dwEmployees.Table = this.dsEmployeeView1.Em
ployees;
```

```
            //
            // dsTaipNe1
            //
            this.dsTaipNe1.DataSetName = "DSTrueFalse";
            this.dsTaipNe1.Locale = new System.Globalization
.CultureInfo("en-US");
            //
            // dsDepartments1
            //
            this.dsDepartments1.DataSetName = "AllDepartment
";
            this.dsDepartments1.Locale = new System.Globaliz
ation.CultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployeeView1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dwEmployees)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsTaipNe1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsDepartments1)).EndInit();

        }
        #endregion


        private void lbInsert_Click(object sender, System.Ev
entArgs e)
        {
            this.Response.Redirect("~/Pages/Inserts/InsertEm
ployee.aspx");
```

```
        }

        private void DataGrid_ItemCommand(object source, Sys
tem.Web.UI.WebControls.DataGridCommandEventArgs e)
        {
            switch (e.CommandName)
            {
                case "Edit":
                    this.Response.Redirect("~/Pages/Inserts/
InsertEmployee.aspx?EmployeeID=" +
                        e.CommandArgument, false);
                    break;

                case "Delete":
                    break;
            }
        }

    }
}
```

ProjectRegistry/Pages/BatchViews/BatchEventView.aspx ProjectRegistry/Pages/BatchViews/BatchEventView.aspx.resx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using ProjectRegistry.Users;

namespace ProjectRegistry.Pages.BatchViews
{
    /// <summary>
    /// Summary description for BatchEventView.
    /// </summary>
    public class BatchEventView : BatchView
    {
        protected ProjectRegistry.DataBase.DataSets.Events.D
SEvents dsBatchEvents1;
        protected System.Web.UI.WebControls.DataGrid dgEvent
s;
        protected System.Data.DataView dwEvents;
        protected System.Web.UI.WebControls.LinkButton lbIns
ert;

        protected BatchEventView() : base
            (new ProjectRegistry.DataBase.Event.EventControl
ler(), "Batch Event View") {}
```

```
        protected override DataView dataView
        {
            get
            {
                return this.dwEvents;
            }
        }



        protected override DataGrid dataGrid
        {
            get
            {
                return this.dgEvents;
            }
        }

        protected override void PRBindData()
        {
            new ProjectRegistry.DataBase.DataSets.Events.Bat
chEventsDSF(this.dsBatchEvents1).fillDataSet();
            base.PRBindData();
        }

        protected override void addFilters() {}

        protected override void managePermissions()
        {
            AccessControl ac =
                new AccessControl(ProjectRegistry.Users.User
.getUser(this.Session));
            for (int i = 0; i < dataGrid.Items.Count; i++)
```

```
            {
                bool canEdit = ac.canEditEvent(dataGrid.Data
Keys[i].ToString());
                // Edit button
                System.Web.UI.Control cntrl = dataGrid.Items
[i].FindControl("lbEdit");
                cntrl.Visible = canEdit;

                // Delete button
                cntrl = dataGrid.Items[i].FindControl("lbDel
ete");
                cntrl.Visible = canEdit;
            }
        }



        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
```

```
        private void InitializeComponent()
        {
            this.dsBatchEvents1 = new ProjectRegistry.DataBa
se.DataSets.Events.DSEvents();
            this.dwEvents = new System.Data.DataView();
            ((System.ComponentModel.ISupportInitialize)(this
.dsBatchEvents1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsBatchEvents1.TableEvents)).BeginInit();
            this.dgEvents.ItemCreated += new System.Web.UI.W
ebControls.DataGridItemEventHandler(this.DataGrid_ItemCreate
d);
            this.dgEvents.ItemCommand += new System.Web.UI.W
ebControls.DataGridCommandEventHandler(this.dgEvents_ItemCom
mand);
            this.dgEvents.PageIndexChanged += new System.Web
.UI.WebControls.DataGridPageChangedEventHandler(this.Paging)
;
            this.dgEvents.SortCommand += new System.Web.UI.W
ebControls.DataGridSortCommandEventHandler(this.Sorting);
            this.dgEvents.ItemDataBound += new System.Web.UI
.WebControls.DataGridItemEventHandler(this.DataGrid_ItemData
Bound);
            this.lbInsert.Click += new System.EventHandler(t
his.lbInsert_Click);
            //
            // dsBatchEvents1
            //
            this.dsBatchEvents1.DataSetName = "dsBatchEvents
";
            this.dsBatchEvents1.Locale = new System.Globaliz
ation.CultureInfo("en-US");
            //
```

```
            // dsBatchEvents1.TableEvents
            //
            this.dsBatchEvents1.TableEvents.Constraints.AddR
ange(new System.Data.Constraint[] {

                                        new System.Data.Unique
Constraint("DSEventsKey1", new string[] {



                                            "ID"}, true)});
            this.dsBatchEvents1.TableEvents.TableName = "Eve
nts";
            //
            // dwEvents
            //
            this.dwEvents.Table = this.dsBatchEvents1.TableE
vents;
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
.dsBatchEvents1.TableEvents)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dwEvents)).EndInit();

        }
        #endregion

        private void dgEvents_ItemCommand(object source, Sys
tem.Web.UI.WebControls.DataGridCommandEventArgs e)
        {
            switch (e.CommandName)
            {
                case "Edit":
```

```
                    this.Response.Redirect("~/Pages/Inserts/
InsertEvent.aspx?EventID=" + e.CommandArgument, false);
                    break;
                case "Delete":
                    ProjectRegistry.DataBase.Event.EventCont
roller pc =
                        (this.Controller as ProjectRegistry.
DataBase.Event.EventController);
                    pc.deleteEvent(e.CommandArgument.ToStrin
g());
                    this.PRBindData();
                    break;
            }
        }

        private void lbInsert_Click(object sender, System.Ev
entArgs e)
        {
            this.Response.Redirect("~/Pages/Inserts/InsertEv
ent.aspx");
        }

    }
}
```

T.1.2 Insert Pages

ProjectRegistry/Pages/Inserts/InsertView.cs

```
using System;

namespace ProjectRegistry.Pages.Inserts
{
    /// <summary>
    /// Details an interface for pages with Insert/Edit logi
cs
    /// </summary>
    public class InsertView : BasePage
    {
        public InsertView(){}
        public InsertView(ProjectRegistry.DataBase.AbstractC
ontroller controller, string id) : base (controller, id)
        {
        }

        /// <summary>
        /// simply inform all controls to do their databindi
ng
        /// </summary>
        protected override void PRBindData()
        {
            this.DataBind();
        }

    }
}
```

ProjectRegistry/Pages/Inserts/InsertTask.aspx ProjectRegistry/Pages/Inserts/InsertTask.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace ProjectRegistry.Pages.Inserts
{
    /// <summary>
    /// Summary description for InsertTask.
    /// </summary>
    public class InsertTask : InsertView
    {
        protected PRControls.FilterDropDownList ddlProjects;

        protected ProjectRegistry.DataBase.DataSets.Projects
.DSProjects dsProjects;
        protected ProjectRegistry.DataBase.DataSets.Employee
s.DSEmployees dsEmployees;
        protected System.Data.DataSet dataSet1;
        protected System.Data.DataTable dataTable1;
        protected System.Data.DataColumn dataColumn1;
        protected System.Web.UI.WebControls.Button btSaveAnd
Continue;
        protected System.Web.UI.WebControls.Button btSave;
        protected System.Web.UI.WebControls.Panel MPanel;
        protected System.Web.UI.WebControls.Panel Panel1;
```

```
        protected ProjectRegistry.DataBase.DataSets.Tasks.DS
Tasks dsTasks1;
        protected System.Web.UI.WebControls.Label lbTaskID;
        protected System.Web.UI.WebControls.Panel pnlSaveBut
tons;
        protected PRControls.DatePicker DatePicker1;
        protected System.Web.UI.WebControls.CompareValidator
 validateProjectSelected;
        protected System.Web.UI.WebControls.DataGrid DataGri
d1;



        public InsertTask():base (new ProjectRegistry.DataBa
se.Task.TaskController(),
            "insert task page") {}

        private string TaskID
        {
            get
            {
                return this.lbTaskID.Text;
            }
            set
            {
                this.lbTaskID.Text = value;
            }
        }

        private string ProjectID;

        protected override void PRBindData()
        {
            new ProjectRegistry.DataBase.DataSets.Projects.P
```

```
rojectsDSF(this.dsProjects).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Employees.
EmployeesDSF(this.dsEmployees).fillDataSet();

            ProjectRegistry.DataBase.DataSets.Tasks.TasksDSF
 tdsf =
                new ProjectRegistry.DataBase.DataSets.Tasks.
TasksDSF(this.dsTasks1);

            if (this.TaskID != "")
            {
                tdsf.fillDataSet(this.Request.Params["TaskID
"].ToString());
                object v = this.dsTasks1.Tables[0].Rows[0].I
temArray.GetValue(11);
                this.ddlProjects.SelectedValue = v.ToString(
);
            }
            else
            {
                for (int i1=0; i1<5; i1++)
                {
                    string[] row = new string[this.dsTasks1.
Tables[0].Columns.Count];
                    row[0] = "" + i1; // ID
                    row[1] = ""; // Project Name
                    row[7] = "false"; // Approved by Project
 Manager
                    row[8] = "false"; // Approved by Departm
ent Manager
                    row[10] = "0"; // Used hours
                    this.dsTasks1.Tables[0].Rows.Add(row);
                }
```

```
            }

            this.DataBind();

            if (ProjectID != null)
            {
                ListItem item = ddlProjects.Items.FindByValu
e(ProjectID);
                ddlProjects.SelectedIndex = ddlProjects.Item
s.IndexOf(item);
            }
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.dsProjects = new ProjectRegistry.DataBase.D
```

```
ataSets.Projects.DSProjects();
            this.dsEmployees = new ProjectRegistry.DataBase.
DataSets.Employees.DSEmployees();
            this.dataSet1 = new System.Data.DataSet();
            this.dataTable1 = new System.Data.DataTable();
            this.dataColumn1 = new System.Data.DataColumn();

            this.dsTasks1 = new ProjectRegistry.DataBase.Dat
aSets.Tasks.DSTasks();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjects)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dataSet1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dataTable1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsTasks1)).BeginInit();
            this.btSaveAndContinue.Click += new System.Event
Handler(this.btSaveAndContinue_Click);
            this.btSave.Click += new System.EventHandler(thi
s.btSave_Click);
            //
            // dsProjects
            //
            this.dsProjects.DataSetName = "DSProjects";
            this.dsProjects.Locale = new System.Globalizatio
n.CultureInfo("en-US");
            //
            // dsEmployees
            //
            this.dsEmployees.DataSetName = "DSEmployees";
```

```
            this.dsEmployees.Locale = new System.Globalizati
on.CultureInfo("en-US");
            //
            // dataSet1
            //
            this.dataSet1.DataSetName = "NewDataSet";
            this.dataSet1.Locale = new System.Globalization.
CultureInfo("lt-LT");
            this.dataSet1.Tables.AddRange(new System.Data.Da
taTable[] {

              this.dataTable1});
            //
            // dataTable1
            //
            this.dataTable1.Columns.AddRange(new System.Data
.DataColumn[] {

                  this.dataColumn1});
            this.dataTable1.TableName = "Table1";
            //
            // dataColumn1
            //
            this.dataColumn1.ColumnName = "Column1";
            //
            // dsTasks1
            //
            this.dsTasks1.DataSetName = "DSTasks";
            this.dsTasks1.Locale = new System.Globalization.
CultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
```

```
.dsProjects)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dataSet1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dataTable1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsTasks1)).EndInit();

        }
        #endregion



        private void InsertTasks()
        {
            ProjectRegistry.DataBase.Task.TaskController tc
=
                this.Controller as ProjectRegistry.DataBase.
Task.TaskController;
            System.Web.UI.WebControls.DataGridItemCollection
 dgic =
                this.DataGrid1.Items;

            // ProjectID
            string projectID = this.ddlProjects.SelectedValu
e;
            if (projectID == "-1") return;
            tc.addParam("ProjectID", projectID);

            for (int i = 0; i < this.DataGrid1.Items.Count;
i++)
            {
```

```
                // Employee ID
                System.Web.UI.WebControls.DropDownList prddl
 =
                    dgic[i].Cells[0].FindControl("ddlVykdyto
jas") as
                    System.Web.UI.WebControls.DropDownList;
                tc.addParam("EmployeeID", prddl.SelectedValu
e);

                // Task name
                System.Web.UI.WebControls.TextBox tb =
                    dgic[i].Cells[1].FindControl("tbName") a
s
                    System.Web.UI.WebControls.TextBox;
                tc.addParam("Name", tb.Text);

                if (tb.Text == "") continue;

                // Planned Hours
                tb = dgic[i].Cells[2].FindControl("tbHours")
 as
                    System.Web.UI.WebControls.TextBox;
                tc.addParam("PlannedHours", tb.Text);

                // StartDate
                tb = dgic[i].Cells[3].FindControl("tbStartDa
te") as
                    System.Web.UI.WebControls.TextBox;
                tc.addParam("StartDate", tb.Text);

                // EndDate
                tb = dgic[i].Cells[4].FindControl("tbEndDate
") as
```

```
                    System.Web.UI.WebControls.TextBox;
                tc.addParam("EndDate", tb.Text);

                tc.addParam("Priority", 2);

                // Description
                tb = dgic[i].Cells[5].FindControl("tbDescrip
tion") as
                    System.Web.UI.WebControls.TextBox;
                tc.addParam("Description", tb.Text);

                tc.addParam("AcceptanceCriteria", "akceptanc
e kraiteria");



                //(this.Controller as ProjectRegistry.DataBa
se.Task.TaskController).insertTask(projectID,employeeID,task
Name);
                if (this.TaskID == "")
                    tc.insertTask();
                else
                    tc.updateTask(this.TaskID);

            }
        }

        private void btSaveAndContinue_Click(object sender,
System.EventArgs e)
        {
            this.InsertTasks();
        }

        protected override void Page_Load(object sender, Sys
```

```
tem.EventArgs e)
        {
            if (this.Page.Request.Params["TaskID"] != null)
                this.TaskID = this.Page.Request.Params["Task
ID"].ToString();
            if (this.Page.Request.Params["ProjectID"] != nul
l)
                this.ProjectID = this.Page.Request.Params["P
rojectID"].ToString();
            base.Page_Load(sender, e);
        }

        private void btSave_Click(object sender, System.Even
tArgs e)
        {
            this.InsertTasks();
            this.Response.Redirect("../BatchViews/BatchTaskV
iew.aspx", false);
        }



    }
}
```

ProjectRegistry/Pages/Inserts/InsertProject.aspx ProjectRegistry/Pages/Inserts/InsertPr

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace ProjectRegistry.Pages.Inserts
{
    public class InsertProject : InsertView
    {
        protected System.Web.UI.WebControls.Button btSave;
        protected System.Web.UI.WebControls.TextBox tbProjec
tID;
        protected System.Web.UI.WebControls.TextBox tbCode;
        protected System.Web.UI.WebControls.TextBox tbName;
        protected System.Web.UI.WebControls.DropDownList ddl
PM;
        protected System.Web.UI.WebControls.TextBox tbClient
;
        protected System.Web.UI.WebControls.HyperLink hlCont
ract;
        protected System.Web.UI.WebControls.TextBox tbClient
Representative;
        protected System.Web.UI.WebControls.TextBox tbBeginD
ate;
        protected System.Web.UI.WebControls.TextBox tbEndDat
e;
```

```
        protected System.Web.UI.WebControls.CheckBox cbFinis
hed;
        protected System.Web.UI.WebControls.TextBox tbPPK;
        protected System.Web.UI.WebControls.TextBox tbProjec
tStaff;
        protected System.Web.UI.WebControls.TextBox tbSize;
        protected System.Web.UI.WebControls.HyperLink hlProj
ectPlan;
        protected System.Web.UI.WebControls.TextBox tbDescri
ption;
        protected ProjectRegistry.DataBase.DataSets.Employee
s.DSEmployees dsEmployees1;
        protected ProjectRegistry.DataBase.DataSets.TrueFals
e.DSTrueFalse dsProjectStatus1;
        protected ProjectRegistry.DataBase.DataSets.Projects
.AllProject dsAllProject1;
        protected System.Web.UI.WebControls.Panel Panel1;
        protected System.Web.UI.WebControls.Panel Panel2;
        protected System.Web.UI.WebControls.Panel Panel3;
        protected System.Web.UI.WebControls.Panel Panel4;
        protected System.Web.UI.WebControls.Panel pnlSaveBut
tons;
        protected System.Web.UI.WebControls.Panel pnlFakeAre
a;
        protected System.Web.UI.HtmlControls.HtmlInputFile f
akebrowse;
        protected System.Web.UI.HtmlControls.HtmlInputFile b
rowse;
        protected System.Web.UI.WebControls.Table DataGrid1;


        public InsertProject():base(new ProjectRegistry.Data
Base.Project.ProjectController(),"InsertProject")
```

```
        {
        }

        /// <summary>
        /// gets PMID from Project Data set table
        /// </summary>
        /// <remarks> can be used after dataBind() only, oth
erwise DS is not filed</remarks>
        private string ProjectManagerID
        {
            get {return this.dsAllProject1.Tables[0].Rows[0]
.ItemArray.GetValue(7).ToString();}
        }

        /// <summary>
        /// gets PMID from Project Data set table
        /// </summary>
        /// <remarks> can be used after dataBind() only, oth
erwise DS is not filed</remarks>
        private string ProjectStatus
        {
            get {return this.dsAllProject1.Tables[0].Rows[0]
.ItemArray.GetValue(5).ToString();}
        }

        private string ProjectID
        {
            get
            {
                return this.tbProjectID.Text;
            }
            set
            {
```

```
                this.tbProjectID.Text=value;
            }
        }

        protected override void PRBindData()
        {
            new ProjectRegistry.DataBase.DataSets.Employees.
EmployeesDSF(this.dsEmployees1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Projects.P
rojectStatusDSF(dsProjectStatus1).fillDataSet();
            if (this.ProjectID!="")
            {
                new ProjectRegistry.DataBase.DataSets.Projec
ts.AllProjectDSF(this.dsAllProject1).fillDataSet(this.Projec
tID);
            }
            else
            {
                string[] row = new string[this.dsAllProject1
.Tables[0].Columns.Count];
                for (int i = 1; i < this.dsAllProject1.Table
s[0].Columns.Count; i++)
                row[i] = "";
                row[0] = "-1";
                row[3] = "12/12/2005";
                row[4] = "12/12/2005";
                row[5] = "false";
                row[7] = "-1";
                row[10] = "0";// size
                this.dsAllProject1.Tables[0].Rows.Add(row);
            }
            this.DataBind();
```

```
            // set up ddlPM to show selected project manager

            if (this.ProjectManagerID!="-1")
                this.ddlPM.SelectedValue=this.ProjectManager
ID;
            this.cbFinished.Checked=System.Convert.ToBoolean
(this.ProjectStatus);




        }
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.dsEmployees1 = new ProjectRegistry.DataBase
.DataSets.Employees.DSEmployees();
            this.dsProjectStatus1 = new ProjectRegistry.Data
```

```
Base.DataSets.TrueFalse.DSTrueFalse();
            this.dsAllProject1 = new ProjectRegistry.DataBas
e.DataSets.Projects.AllProject();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectStatus1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsAllProject1)).BeginInit();
            this.btSave.Click += new System.EventHandler(thi
s.btSave_Click);
            //
            // dsEmployees1
            //
            this.dsEmployees1.DataSetName = "DSEmployees";
            this.dsEmployees1.Locale = new System.Globalizat
ion.CultureInfo("en-US");
            //
            // dsProjectStatus1
            //
            this.dsProjectStatus1.DataSetName = "DSProjectSt
atus";
            this.dsProjectStatus1.Locale = new System.Global
ization.CultureInfo("en-US");
            //
            // dsAllProject1
            //
            this.dsAllProject1.DataSetName = "AllProject";
            this.dsAllProject1.Locale = new System.Globaliza
tion.CultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
```

```
.dsEmployees1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectStatus1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsAllProject1)).EndInit();

        }
        #endregion



        // inserts or updates a project
        private void btSave_Click(object sender, System.Even
tArgs e)
        {
            if (this.Page.IsValid==false) return;
            ProjectRegistry.DataBase.Project.ProjectControll
er c=this.Controller as ProjectRegistry.DataBase.Project.Pro
jectController;

            c.addParam("Code",this.tbCode.Text);
            c.addParam("Name",this.tbName.Text);
            c.addParam("BeginDate",this.tbBeginDate.Text);
            c.addParam("EndDate",this.tbEndDate.Text);
            if (this.cbFinished.Checked)
                c.addParam("Finished",1); else
                c.addParam("Finished",0);
            c.addParam("Description",this.tbDescription.Text
);
            c.addParam("ProjectManagerID",this.ddlPM.Selecte
dValue);
            c.addParam("ProjectTeam",this.tbProjectStaff.Tex
t);
            c.addParam("PPK",this.tbPPK.Text);
```

```
            c.addParam("ProjectSize",this.tbSize.Text);
            c.addParam("Client",this.tbClient.Text);
            c.addParam("ClientRepresentative",this.tbClientR
epresentative.Text);
            c.addParam("Contract",this.hlContract.NavigateUr
l);
            c.addParam("ProjectPlan",this.hlProjectPlan.Navi
gateUrl);

            // if ProjectID is set, then we are in Edit Mode

            if (this.ProjectID!="")
            {
                c.updateProject(this.ProjectID);
                this.Response.Redirect("../BatchViews/BatchP
rojectView.aspx",false);
                return;
            }
            else // we are in insert mode
            {
                c.addProject();
                this.Response.Redirect("../BatchViews/BatchP
rojectView.aspx",false);
            }
        }

        private void CustomValidator2_ServerValidate(object
source, System.Web.UI.WebControls.ServerValidateEventArgs ar
gs)
        {
        }

        private void sqlConnection2_InfoMessage(object sende
```

```
r, System.Data.SqlClient.SqlInfoMessageEventArgs e)
        {
        }

        protected override void Page_Load(object sender, Sys
tem.EventArgs e)
        {
            if (this.Page.Request.Params["ProjectID"]!=null)

                this.ProjectID=this.Page.Request.Params["Pro
jectID"];
            base.Page_Load(sender,e);
        }

    }
}
```

ProjectRegistry/Pages/Inserts/InsertEvent.aspx ProjectRegistry/Pages/Inserts/InsertEve

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace ProjectRegistry.Pages.Inserts
{

    public class InsertEvent : InsertView
    {
        protected ProjectRegistry.DataBase.DataSets.Events.D
SEvents dsBatchEvents1;
        protected System.Web.UI.WebControls.Label lbEventID;



        protected System.Web.UI.WebControls.Table DataGrid1;

        protected System.Web.UI.WebControls.LinkButton LinkB
utton1;

        protected System.Web.UI.WebControls.HyperLink hlLink
ToReport;
        protected PRControls.FilterDropDownList       ddlPro
jectID;
        protected System.Web.UI.WebControls.TextBox   tbName
;
```

```
        protected ProjectRegistry.DataBase.DataSets.Projects
.dsProjectCode dsProjectCode1;
        protected System.Web.UI.WebControls.Panel SavePanel;

        protected System.Web.UI.WebControls.Panel MainPanel;

        protected System.Web.UI.WebControls.TextBox   tbDate
;

        public InsertEvent() :base (new ProjectRegistry.Data
Base.Event.EventController(),"Insert Task")
        {
        }

        protected override void Page_Load(object sender, Sys
tem.EventArgs e)
        {
            if (this.Request.Params["EventID"] != null)
                this.EventID = this.Request.Params["EventID"
];
            base.Page_Load(sender,e);
        }

        private string EventID
        {
            get
            {
                return this.lbEventID.Text;
            }
            set
            {
                this.lbEventID.Text=value;
            }
```

```
        }

        protected override void PRBindData()
        {
            new ProjectRegistry.DataBase.DataSets.Projects.P
rojectCodeDSF(this.dsProjectCode1).fillDataSet();
            ProjectRegistry.DataBase.DataSets.Events.BatchEv
entsDSF bedsf=new ProjectRegistry.DataBase.DataSets.Events.B
atchEventsDSF(this.dsBatchEvents1);
            if (this.EventID != "")
                bedsf.fillDataSet(this.EventID);
            else
            {
                bedsf.fillDataSetEmpty();
            }

            base.PRBindData ();

            if (bedsf.getFromDS("ProjectID") != "-1")
                this.ddlProjectID.SelectedValue = bedsf.getF
romDS("ProjectID").ToString();
        }


        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
```

```
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.dsBatchEvents1 = new ProjectRegistry.DataBa
se.DataSets.Events.DSEvents();
            this.dsProjectCode1 = new ProjectRegistry.DataBa
se.DataSets.Projects.dsProjectCode();
            ((System.ComponentModel.ISupportInitialize)(this
.dsBatchEvents1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsBatchEvents1.TableEvents)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectCode1)).BeginInit();
            this.LinkButton1.Click += new System.EventHandle
r(this.LinkButton1_Click);
            //
            // dsBatchEvents1
            //
            this.dsBatchEvents1.DataSetName = "dsBatchEvents
";
            this.dsBatchEvents1.Locale = new System.Globaliz
ation.CultureInfo("en-US");
            //
            // dsBatchEvents1.TableEvents
            //
            this.dsBatchEvents1.TableEvents.Constraints.AddR
```

```
ange(new System.Data.Constraint[] {

                                        new System.Data.Unique
Constraint("DSEventsKey1", new string[] {



                                            "ID"}, true)});
            this.dsBatchEvents1.TableEvents.TableName = "Eve
nts";
            //
            // dsProjectCode1
            //
            this.dsProjectCode1.DataSetName = "dsProjectCode
";
            this.dsProjectCode1.Locale = new System.Globaliz
ation.CultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
.dsBatchEvents1.TableEvents)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsBatchEvents1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectCode1)).EndInit();

        }
        #endregion

        private void LinkButton1_Click(object sender, System
.EventArgs e)
        {
            ProjectRegistry.DataBase.Event.EventController c
 = this.Controller as ProjectRegistry.DataBase.Event.EventCo
```

```
ntroller;

            c.addParam("ProjectID",this.ddlProjectID.Selecte
dValue);
            c.addParam("Name",this.tbName.Text);
            c.addParam("Date",this.tbDate.Text);
            c.addParam("LinkToReport", this.hlLinkToReport.N
avigateUrl);

            if (this.EventID != "")
            { // this is edit mode
                c.updateEvent(this.EventID);
            }
            else
            { // insert mode
                c.addEvent();
            }
            this.Response.Redirect("~/Pages/BatchViews/Batch
EventView.aspx");
        }
    }
}
```

ProjectRegistry/Pages/Inserts/InsertEmployee.aspx ProjectRegistry/Pages/Inserts/InsertEmployee.aspx.resx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace ProjectRegistry.Pages.Inserts
{
    /// <summary>
    /// Summary description for InsertEmployee.
    /// </summary>
    public class InsertEmployee : InsertView
    {
        protected ProjectRegistry.DataBase.DataSets.Employee
s.DSEmployees dsEmployees1;
        protected System.Web.UI.WebControls.Label lbEmployee
ID;

        protected ProjectRegistry.DataBase.DataSets.Departme
nts.AllDepartment dsAllDepartment1;

        protected System.Web.UI.WebControls.Table DataGrid1;



        protected System.Web.UI.WebControls.CheckBox cbWorki
ng;
```

```
        protected System.Web.UI.WebControls.TextBox tbName;
        protected System.Web.UI.WebControls.TextBox tbEmail;

        protected System.Web.UI.WebControls.TextBox tbWindow
sLogin;
        protected PRControls.FilterDropDownList ddlDepartmen
ts;
        protected System.Web.UI.WebControls.Panel MainPanel;

        protected System.Web.UI.WebControls.Panel SaveButton
Panel;
        protected System.Web.UI.WebControls.LinkButton btnSa
ve;
        protected System.Web.UI.WebControls.CheckBox cbAdmin
istrator;

        public InsertEmployee() : base(new ProjectRegistry.D
ataBase.Employee.EmployeeController(), "InsertEmployee") {}



        protected string EmployeeID
        {
            get
            {
                return this.lbEmployeeID.Text;
            }
            set
            {
                this.lbEmployeeID.Text = value;
            }
        }

        protected override void Page_Load(object sender, Sys
```

```
tem.EventArgs e)
        {
            if (this.Request.Params["EmployeeID"] != null)
                this.EmployeeID = this.Request.Params["Emplo
yeeID"].ToString();
            base.Page_Load(sender, e);
            // Put user code to initialize the page here
        }

        protected override void PRBindData()
        {
            ProjectRegistry.DataBase.DataSets.Employees.Empl
oyeesDSF edsf
                = new ProjectRegistry.DataBase.DataSets.Empl
oyees.EmployeesDSF(this.dsEmployees1);
            if (this.EmployeeID != "")
            {
                edsf.fillDataSet(this.EmployeeID);
                btnSave.Text = "Issaugoti";
            }
            else
                edsf.fillDataSetEmpty();
            new ProjectRegistry.DataBase.DataSets.Department
s.AllDepartmentDSF(this.dsAllDepartment1).fillDataSet();

            base.PRBindData();

            if (edsf.getFromDS("DepartmentID") != "-1")
                this.ddlDepartments.SelectedValue = edsf.get
FromDS("DepartmentID");
        }

        #region Web Form Designer generated code
```

```
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.dsEmployees1 = new ProjectRegistry.DataBase
.DataSets.Employees.DSEmployees();
            this.dsAllDepartment1 = new ProjectRegistry.Data
Base.DataSets.Departments.AllDepartment();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsAllDepartment1)).BeginInit();
            this.btnSave.Command += new System.Web.UI.WebCon
trols.CommandEventHandler(this.btSave_Click);
            //
            // dsEmployees1
            //
            this.dsEmployees1.DataSetName = "DSEmployees";
            this.dsEmployees1.Locale = new System.Globalizat
```

```
ion.CultureInfo("en-US");
            //
            // dsAllDepartment1
            //
            this.dsAllDepartment1.DataSetName = "AllDepartme
nt";
            this.dsAllDepartment1.Locale = new System.Global
ization.CultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsAllDepartment1)).EndInit();

        }
        #endregion

        // inserts or updates an employee
        private void btSave_Click(object sender, System.Web.
UI.WebControls.CommandEventArgs e)
        {
            if (!this.Page.IsValid) return;
            ProjectRegistry.DataBase.Employee.EmployeeContro
ller c =
                this.Controller as ProjectRegistry.DataBase.
Employee.EmployeeController;

            c.addParam("Name", this.tbName.Text);
            c.addParam("Email", this.tbEmail.Text);
            c.addParam("Working", this.cbWorking.Checked? 1
: 0);
            c.addParam("Administrator", this.cbAdministrator
```

```
.Checked? 1 : 0);
            c.addParam("DepartmentID", this.ddlDepartments.S
electedValue);
            c.addParam("WindowsLogin", this.tbWindowsLogin.T
ext);



            if (this.EmployeeID == "")
                // we are in insert mode
                c.insertEmployee();
            else
                c.updateEmployee(this.EmployeeID);
            this.Response.Redirect("~/Pages/BatchViews/Batch
EmployeeView.aspx", false);
        }

    }
}
```

T.1.3 Edit Pages

ProjectRegistry/Pages/Edits/EditView.cs

```
using System;
using System.Data;

using System.Web.UI.WebControls;
using ProjectRegistry.Pages;
using ProjectRegistry.DataBase;

namespace ProjectRegistry.Pages.Edits
{
    /// <summary>
    /// Summary description for EditView.
    /// </summary>
    public class EditView : BasePage
    {
        public EditView(AbstractController controller,string
 id ) : base (controller,id) {}
        public EditView() : base () {}

        /*protected override void BindData()
        {
            object key = Session["key"];
            if (key != null)
            {
                controller.selectByKey(dataView, key.ToStrin
g());
                DataBind();
            }
            //TODO - redo all edit page
        }*/

    }
```

```
}
```

ProjectRegistry/Pages/Edits/EditProject.aspx ProjectRegistry/Pages/Edits/EditProject.a

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using ProjectRegistry.DataBase.Project;

namespace ProjectRegistry.Pages.Edits
{
    /// <summary>
    /// Summary description for EditProject.
    /// </summary>
    public class EditProject : EditView
    {
        protected System.Web.UI.WebControls.Table Table2;
        protected System.Web.UI.WebControls.Label Label5;
        protected System.Web.UI.WebControls.Label Label1;
        protected System.Web.UI.WebControls.Button btSave;
        //protected System.Web.UI.WebControls.DropDownList d
dlPM;
        protected ProjectRegistry.DataBase.DataSets.Employee
s.DSEmployees dsEmployees1;
        protected System.Web.UI.WebControls.DropDownList Dro
pDownList1;
        protected System.Web.UI.WebControls.Label Label3;
        protected System.Web.UI.WebControls.TextBox TextBox1
```

```
;
        //protected System.Web.UI.WebControls.DropDownList d
dlPMs;
        protected ProjectRegistry.DataBase.DataSets.Projects
.DSProjects dsProjects1;

        public EditProject() : base(new ProjectController(),
"EditProject") {}

        /*protected override DataSet dataSet
        {
            get { return dsProjectView; }
        }*/

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
```

```
            this.dsProjects1 = new ProjectRegistry.DataBase.
DataSets.Projects.DSProjects();
            this.dsEmployees1 = new ProjectRegistry.DataBase
.DataSets.Employees.DSEmployees();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjects1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees1)).BeginInit();
            this.btSave.Click += new System.EventHandler(thi
s.btSave_Click);
            //
            // dsProjects1
            //
            this.dsProjects1.DataSetName = "DSProjects";
            this.dsProjects1.Locale = new System.Globalizati
on.CultureInfo("en-US");
            //
            // dsEmployees1
            //
            this.dsEmployees1.DataSetName = "DSEmployees";
            this.dsEmployees1.Locale = new System.Globalizat
ion.CultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjects1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees1)).EndInit();

        }
        #endregion

        // gets the DropDownList Project Manager from the ht
```

```
ml table.
        private System.Web.UI.WebControls.DropDownList ddlPM

        {
            get{return this.FindControl("ddlPM") as System.W
eb.UI.WebControls.DropDownList;}
        }

        private System.Web.UI.WebControls.TextBox tbCode
        {
            get{return this.FindControl("tbCode") as System.
Web.UI.WebControls.TextBox;}
        }

        private System.Web.UI.WebControls.TextBox tbProjectI
D
        {
            get{return this.FindControl("tbProjectID") as Sy
stem.Web.UI.WebControls.TextBox;}
        }

        private System.Web.UI.WebControls.TextBox tbName
        {
            get{return this.FindControl("tbName") as System.
Web.UI.WebControls.TextBox;}
        }

        private string ProjectID
        {
            get
            {
                if (this.Page.Request.Params["ProjectID"]==n
ull)
```

```
                    throw new System.Exception("must pass Pr
ojectID parameter");
                return this.Page.Request.Params["ProjectID"]
.ToString();
            }
        }

        private string ProjectManagerID
        {
            get {return this.dsProjects1.Tables[0].Rows[0].I
temArray.GetValue(8).ToString();}
        }

        protected override void BindData()
        {
            //if (this.Request.Params["ProjectID"]==null) th
row new System.Exception("must provide ProjectID param");

            //string projectID=this.Request.Params["ProjectI
D"];
            new ProjectRegistry.DataBase.DataSets.Employees.
EmployeesDSF(this.dsEmployees1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Projects.P
rojectsDSF(this.dsProjects1).fillDataSet(this.ProjectID);
            // force all controls to bind to data
            this.DataBind();

            // set-up ddlPM to show selected project manager

            this.ddlPM.SelectedValue=this.ProjectManagerID;
        }

        private void btSave_Click(object sender, System.Even
```

```
tArgs e)
        {
            ProjectRegistry.DataBase.Project.ProjectControll
er c=this.Controller as ProjectRegistry.DataBase.Project.Pro
jectController;

            string pmid=this.ddlPM.SelectedValue;
            string code=this.tbCode.Text;
            string name=this.tbName.Text;
            string projectid=this.tbProjectID.Text;
            c.updateProject(projectid,code,name,pmid);
            this.Response.Redirect("../BatchViews/BatchProje
ctView.aspx",false);
        }



    }
}
```

T.1.4 DetailView Pages

ProjectRegistry/Pages/DetailView/ViewProject.aspx ProjectRegistry/Pages/DetailView/Vie

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace ProjectRegistry.Pages.BatchViews
{
    /// <summary>
    /// Summary description for ViewProject.
    /// </summary>
    public class ViewProject : BasePage
    {
        protected System.Web.UI.WebControls.HyperLink hlTask
s;
        protected ProjectRegistry.DataBase.DataSets.Projects
.AllProject dsAllProject1;
        protected System.Web.UI.WebControls.HyperLink hlEven
ts;

        protected System.Web.UI.WebControls.Label lbProjectI
D;
        protected System.Web.UI.WebControls.Label lbCode;
        protected System.Web.UI.WebControls.Label lbName;
        protected System.Web.UI.WebControls.Label lbDescript
ion;
```

```
        protected System.Web.UI.WebControls.DropDownList ddl
PM;
        protected System.Web.UI.WebControls.Label lbClient;
        protected System.Web.UI.WebControls.HyperLink hlCont
ract;
        protected System.Web.UI.WebControls.Label lbClientRe
presentative;
        protected System.Web.UI.WebControls.Label lbBeginDat
e;
        protected System.Web.UI.WebControls.Label lbEndDate;

        protected System.Web.UI.WebControls.CheckBox cbFinis
hed;
        protected System.Web.UI.WebControls.Label lbPPK;
        protected System.Web.UI.WebControls.Label lbProjectS
taff;
        protected System.Web.UI.WebControls.Label lbSize;
        protected System.Web.UI.WebControls.HyperLink hlProj
ectPlan;
        protected System.Web.UI.WebControls.Label lbPMName;
        protected System.Web.UI.WebControls.Label lbFinished
;
        protected ProjectRegistry.DataBase.DataSets.Employee
s.DSEmployees dsEmployees1;
        protected ProjectRegistry.DataBase.DataSets.TrueFals
e.DSTrueFalse dsProjectStatus1;
        protected System.Web.UI.WebControls.Panel SavePanel;

        protected System.Web.UI.WebControls.Panel MainPanel;

        protected System.Web.UI.WebControls.Table Table1;
```

```
        private string ProjectID
        {
            get
            {
                return this.lbProjectID.Text;
            }
            set
            {
                this.lbProjectID.Text=value;
            }
        }



        protected override void PRBindData()
        {
            // fill data set
            //new ProjectRegistry.DataBase.DataSets.Projects
.ProjectsDSF(this.dsProjects1).fillDataSet(this.ProjectID);
            new ProjectRegistry.DataBase.DataSets.Employees.
EmployeesDSF(this.dsEmployees1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Projects.P
rojectStatusDSF(dsProjectStatus1).fillDataSet();
            new ProjectRegistry.DataBase.DataSets.Projects.A
llProjectDSF(this.dsAllProject1).fillDataSet(this.ProjectID)
;
            // wait till all controls bind their data
            this.DataBind();
            if (System.Convert.ToBoolean(this.dsAllProject1.
Tables[0].Rows[0].ItemArray.GetValue(5)) == false)
                this.lbFinished.Text="Ne"; else
                this.lbFinished.Text="Taip";

            // then find ProjectName and build URLs for two
```

```
HL controls
            //string projectName=(this.FindControl("lbName")
as System.Web.UI.WebControls.Label).Text;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.dsAllProject1 = new ProjectRegistry.DataBas
e.DataSets.Projects.AllProject();
            this.dsEmployees1 = new ProjectRegistry.DataBase
.DataSets.Employees.DSEmployees();
            this.dsProjectStatus1 = new ProjectRegistry.Data
Base.DataSets.TrueFalse.DSTrueFalse();
            ((System.ComponentModel.ISupportInitialize)(this
.dsAllProject1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
```

```
.dsEmployees1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectStatus1)).BeginInit();
            //
            // dsAllProject1
            //
            this.dsAllProject1.DataSetName = "AllProject";
            this.dsAllProject1.Locale = new System.Globaliza
tion.CultureInfo("en-US");
            //
            // dsEmployees1
            //
            this.dsEmployees1.DataSetName = "DSEmployees";
            this.dsEmployees1.Locale = new System.Globalizat
ion.CultureInfo("en-US");
            //
            // dsProjectStatus1
            //
            this.dsProjectStatus1.DataSetName = "DSProjectSt
atus";
            this.dsProjectStatus1.Locale = new System.Global
ization.CultureInfo("en-US");
            this.Load += new System.EventHandler(this.Page_L
oad);
            ((System.ComponentModel.ISupportInitialize)(this
.dsAllProject1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsEmployees1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this
.dsProjectStatus1)).EndInit();

        }
        #endregion
```

```
        protected override void Page_Load(object sender, Sys
tem.EventArgs e)
        {
            //this.ProjectID="";
            if (this.Page.Request.Params["ProjectID"] != nul
l)
                this.ProjectID = this.Page.Request.Params["P
rojectID"];
            base.Page_Load(sender, e);
        }

    }
}
```

T.1.5 Administrator Pages

ProjectRegistry/Pages/Administrator/Administrate.aspx ProjectRegistry/Pages/Administrator/Administrate.aspx.resx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace ProjectRegistry.Pages.BatchViews
{
    /// <summary>
    /// Summary description for Administrate.
    /// </summary>
    public class Administrate : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.Panel Panel1;
        protected System.Web.UI.WebControls.Panel MainPanel;

        protected System.Web.UI.WebControls.LinkButton lbEmp
loyees;

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
```

```
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not mo
dify
        /// the contents of this method with the code editor
.
        /// </summary>
        private void InitializeComponent()
        {
            this.lbEmployees.Click += new System.EventHandle
r(this.lbEmployees_Click);

        }
        #endregion

        private void lbEmployees_Click(object sender, System
.EventArgs e)
        {
            this.Response.Redirect("~/Pages/BatchViews/Batch
EmployeeView.aspx",false);
        }
    }
}
```

T.2 Users Management

ProjectRegistry/Users/AccessControl.cs

```
using System;
using ProjectRegistry.DataBase.DataSets.Events;
using ProjectRegistry.DataBase.DataSets.Tasks;
using ProjectRegistry.DataBase.DataSets.Projects;

namespace ProjectRegistry.Users
{
    /// <summary>
    /// Summary description for AccessControl.
    /// </summary>
    public class AccessControl
    {

        public AccessControl(User user)
        {
            this.user = user;
        }

        public bool canEditTask(string TaskID)
        {
            if (isTaskExecutor(TaskID))
                return true;

            DSTasks task = getTaskDescription(TaskID);
            int project = task.Tables[0].Columns.IndexOf("Pr
ojectID");
            if (isProjectManager(task.Tables[0].Rows[0].Item
Array[project].ToString()))
                return true;
            return false;
```

```
        }

        public bool canEditEvent(string EventID)
        {
            DSEvents Event = getEventDescription(EventID);
            int project = Event.Tables[0].Columns.IndexOf("P
rojectID");
            if (isProjectManager(Event.Tables[0].Rows[0].Ite
mArray[project].ToString()))
                return true;
            return false;
        }

        /// <summary>
        /// Checks whether the user is manager of a project
        /// </summary>
        public bool isProjectManager(string ProjectID)
        {
            DSProjects project = getProjectDescription(Proje
ctID);
            int PMColumn = project.Tables[0].Columns.IndexOf
("PMID");
            return project.Tables[0].Rows[0].ItemArray[PMCol
umn].ToString() ==
                user.UserID.ToString();
        }

        public bool isTaskExecutor(string TaskID)
        {
            DSTasks task = getTaskDescription(TaskID);
            int EmployeeColumn = task.Tables[0].Columns.Inde
xOf("EmployeeID");
            return task.Tables[0].Rows[0].ItemArray[Employee
```

```
Column].ToString() ==
                user.UserID.ToString();
        }

        private DSEvents getEventDescription(string EventID)

        {
            if ((eventDescription == null) || (EventID != th
is.EventID))
            {
                eventDescription = new DSEvents();
                new BatchEventsDSF(eventDescription).fillDat
aSet(EventID);
                this.EventID = EventID;
            }
            return eventDescription;
        }

        private DSEvents eventDescription;
        private string EventID;

        private DSTasks getTaskDescription(string TaskID)
        {
            if ((taskDescription == null) || (TaskID != this
.TaskID))
            {
                taskDescription = new DSTasks();
                new TasksDSF(taskDescription).fillDataSet(Ta
skID);
                this.TaskID = TaskID;
            }
            return taskDescription;
        }
```

```
        private DSTasks taskDescription;
        private string TaskID;

        private DSProjects getProjectDescription(string Proj
ectID)
        {
            if ((projectDescription == null) || (this.Projec
tID != ProjectID))
            {
                projectDescription = new DSProjects();
                new ProjectsDSF(projectDescription).fillData
Set(ProjectID);
                this.ProjectID = ProjectID;
            }
            return projectDescription;
        }

        private DSProjects projectDescription;
        private string ProjectID;

        private User user;

    }
}
```

ProjectRegistry/Users/User.cs

```
using System;
using System.Data;

using ProjectRegistry.DataBase.DataSets.Employees;

namespace ProjectRegistry.Users
{
    /// <summary>
    /// Summary description for User.
    /// </summary>
    public class User
    {
        private int userID = -1;

        public int UserID
        {
            get
            {
                return userID;
            }
        }

        public static string UserDomainName
        {
            get
            {
                return System.Security.Principal.WindowsIden
tity.GetCurrent().Name;
            }
        }

        private bool isAdministrator;
```

```
        public bool IsAdministrator
        {
            get
            {
                return isAdministrator;
            }
        }

        private User()
        {
            DSEmployees users = new DSEmployees();
            new EmployeesDSF(users).fillDataSet();
            int logIn = users.Tables[0].Columns.IndexOf("Win
dowsLogIn");

            string CurrentUser = User.UserDomainName;
            foreach (DataRow user in users.Tables[0].Rows)
                if (user.ItemArray[logIn].Equals(CurrentUser
))
                {
                    int id = users.Tables[0].Columns.IndexOf
("ID");
                    userID = Convert.ToInt32(user.ItemArray[
id]);

                    int admin = users.Tables[0].Columns.Inde
xOf("Administrator");
                    isAdministrator = (user.ItemArray[admin]
.ToString() == "True");
                    return;
                }
        }
```

```
        public static User getUser(System.Web.SessionState.H
ttpSessionState Session)
        {
            if (Session["UserInfo"] == null)
                Session["UserInfo"] = new ProjectRegistry.Us
ers.User();
            return Session["UserInfo"] as ProjectRegistry.Us
ers.User;
        }
    }
}
```

sectionPictures

ProjectRegistry/Pictures/calendar.gif

ProjectRegistry/Pictures/Delete.gif

ProjectRegistry/Pictures/Edit.gif

ProjectRegistry/Pictures/logo_bb.gif

ProjectRegistry/Pictures/Tasks.gif

ProjectRegistry/Pictures/View.gif

sectionHeader and Footer Controls

ProjectRegistry/HeaderControl/CopyrightControl.ascx

ProjectRegistry/HeaderControl/CopyrightControl.ascx.resx

ProjectRegistry/HeaderControl/CopyrightControl.ascx.cs

```
namespace ProjectRegistry
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    ///     Summary description for WebUserControl1.
    /// </summary>
    public class WebUserControl1 : System.Web.UI.UserControl

    {
        protected System.Web.UI.WebControls.Panel pnlCopyRig
ht;
        protected System.Web.UI.WebControls.ImageButton Imag
eButton1;

        private void Page_Load(object sender, System.EventAr
gs e)
```

```
        {
            // Put user code to initialize the page here
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        ///     Required method for Designer support - do no
t modify
        ///     the contents of this method with the code ed
itor.
        /// </summary>
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_L
oad);

        }
        #endregion
    }
}
```

ProjectRegistry/HeaderControl/HeaderControl.ascx

ProjectRegistry/HeaderControl/HeaderControl.ascx.resx

ProjectRegistry/HeaderControl/HeaderControl.ascx.cs

```
namespace Project_Register.Clean
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    ///     Summary description for HeaderControl.
    /// </summary>
    public class HeaderControl : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.ImageButton Imag
eButton1;
        protected System.Web.UI.WebControls.Label Label7;
        protected System.Web.UI.WebControls.ImageButton Imag
ebutton1;
        protected System.Web.UI.WebControls.Label HeaderCtrl
Date;
        protected System.Web.UI.WebControls.Label Label6;

        protected System.Web.UI.WebControls.HyperLink hlProj
ects;
        protected System.Web.UI.WebControls.HyperLink hlMyPr
ojects;
        protected System.Web.UI.WebControls.HyperLink hlMyTa
sks;
        protected System.Web.UI.WebControls.HyperLink hlTask
```

```
s;
        protected System.Web.UI.WebControls.HyperLink hlMyAp
provals;
        protected System.Web.UI.WebControls.HyperLink hlEven
ts;
        protected System.Web.UI.WebControls.HyperLink hlRepo
rts;
        protected System.Web.UI.WebControls.Image Image1;
        protected System.Web.UI.WebControls.Panel pnlHeader;

        protected System.Web.UI.WebControls.Panel pnlLogo;
        protected System.Web.UI.WebControls.Table pblHeaderM
enu;
        protected System.Web.UI.WebControls.HyperLink hlAdmi
nistration;
        protected System.Web.UI.WebControls.TableCell AdminP
age;

        private void Page_Load(object sender, System.EventAr
gs e)
        {
            ProjectRegistry.Users.User user = ProjectRegistr
y.Users.User.getUser(this.Session);
            this.hlMyProjects.NavigateUrl = "~/Pages/BatchVi
ews/BatchProjectView.aspx?PMID=" + user.UserID;
            this.hlTasks.NavigateUrl = "~/Pages/BatchViews/B
atchTaskView.aspx?ProjectID=-1&EmployeeID=-1";
            this.hlMyTasks.NavigateUrl = "~/Pages/BatchViews
/BatchTaskView.aspx?EmployeeID=" + user.UserID;
            this.hlEvents.NavigateUrl = "~/Pages/BatchViews/
BatchEventView.aspx";
            this.AdminPage.Visible = user.IsAdministrator;
        }
```

```
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
 Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        ///     Required method for Designer support - do no
t modify
        ///     the contents of this method with the code ed
itor.
        /// </summary>
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_L
oad);

        }
        #endregion
    }
}
```

sectionDatabase

ProjectRegistry/DataBase/AbstractController.cs

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ProjectRegistry.DataBase
{
    /// <summary>
    /// Controllers provides methods of adding, removing, ch
anging data in DB.
    /// each controller manipulates his own logical ara, pro
viding custom methods.
    /// </summary>
    public abstract class AbstractController
    {
        protected ProjectRegistry.DataBase.DBManager dbManag
er;

        public AbstractController()
        {
            this.dbManager = new DBManager();
        }

        public void addParam(string parameterName,object par
ameterValue)
        {
            this.dbManager.setSqlParam(parameterName,paramet
erValue);
        }
    }
}
```

ProjectRegistry/DataBase/DBManager.cs

```
using System;
using System.Data.SqlClient;

namespace ProjectRegistry.DataBase
{
    /// <summary>
    /// Base object designed to accomplishe base tasks with
database:
    /// 1. Connect to it
    /// 2. Execute a query
    /// </summary>
    public class DBManager
    {
        private System.Data.SqlClient.SqlConnection connecti
on;

        private System.Data.SqlClient.SqlCommand     command
;
        private System.Data.SqlClient.SqlDataAdapter dataAda
pter;

        public DBManager()
        {
            // A connection with datase is being established

            connection = new System.Data.SqlClient.SqlConnec
tion();
            connection.ConnectionString = connectionString;
        }

        public void executeSQLNonQuery(string query)
        {
```

```
            this.sqlCommand.CommandText=query;
            this.sqlCommand.Connection.Open();
            this.sqlCommand.ExecuteNonQuery();
            this.sqlCommand.Connection.Close();
        }

        public System.Data.SqlClient.SqlConnection sqlConnec
tion
        {
            get
            {
                return this.connection;
            }
        }

        public SqlCommand sqlCommand
        {
            get
            {
                if (this.command == null)
                {
                    this.command = new SqlCommand();
                    this.command.CommandType = System.Data.C
ommandType.Text;
                    this.command.Connection = this.connectio
n;
                }
                return this.command;
            }
        }

        public SqlDataAdapter sqlAdapter
```

```
        {
            get
            {
                if (this.dataAdapter==null)
                {
                    this.dataAdapter=new SqlDataAdapter(this
.sqlCommand);
                }
                return this.dataAdapter;
            }
        }



        /// <summary>
        /// returns a connection string, based on a user dom
ain name
        /// </summary>
        protected string connectionString
        {
            get
            {
                switch (ProjectRegistry.Users.User.UserDomai
nName)
                {
                    case @"ARKLYS\ASPNET":
                        return @"workstation id=Integrated S
ecurity=SSPI;Packet Size=4096;Initial Catalog=pr;Persist Sec
urity Info=False;Workstation ID=ARKLYS;";
                    case @"VIKTORAS\ASPNET":
                        return @"workstation id=VIKTORAS;pac
ket size=4096;integrated security=SSPI;data source=""VIKTORA
S\VIKTORAS"";persist security info=False;initial catalog=pr"
;
```

```
                    case @"RAGANYNAS\ASPNET":
                        return @"Integrated Security=SSPI;In
itial Catalog=pr;Data Source=RAGANYNAS;Packet Size=4096;Work
station ID=RAGANYNAS;";
                    case @"PROGRAM1\ASPNET":
                        return @"workstation id=PROGRAM1;pac
ket size=4096;integrated security=SSPI;data source=""PROGRAM
1\PRDEMOSERV"";initial catalog=pr";
                    case @"MINDAUGASV\ASPNET":
                        return @"Integrated Security=SSPI;In
itial Catalog=pr;Data Source=MINDAUGASV\MINDAUGASV;Packet Si
ze=4096;Workstation ID=MINDAUGASV;";
                    default:
                        throw new System.Exception("getConne
ctionString returned NULL.");

                }
            }
        }



        /// <summary>
        /// Sets parameters for the upcomming SQL query
        /// </summary>
        public void setSqlParam(string parameterName, object
 value)
        {
            parameterName="@" + parameterName;
            if (this.sqlCommand.Parameters.Contains(paramete
rName))
                this.sqlCommand.Parameters.RemoveAt(paramete
rName);
            this.sqlCommand.Parameters.Add(parameterName, va
```

```
lue);
        }
    }
}
```

subsectionDatabase Controllers

ProjectRegistry/DataBase/Employee/EmployeeController.cs

```
using System;

namespace ProjectRegistry.DataBase.Employee
{
    /// <summary>
    /// Handles data manipulation with Emploee table
    /// </summary>
    public class EmployeeController:AbstractController
    {
        public EmployeeController() : base () {}

        public void insertEmployee()
        {
            string query = @"INSERT INTO Employees
                        (Name, Email, Working, DepartmentID,
 WindowsLogin, Administrator)
                        VALUES
                        (@Name, @Email, @Working, @Departmen
tID, @WindowsLogin, @Administrator)";
            this.dbManager.executeSQLNonQuery(query);
        }

        public void updateEmployee(string id)
        {
            this.addParam("EmployeeID", id);
            string query = @"UPDATE Employees SET
                        Name=@Name, Email=@Email, Working=@W
orking, DepartmentID=@DepartmentID,
                        WindowsLogin=@WindowsLogin, Administ
rator=@Administrator
                        Where Employees.ID = '" + id + "'";
```

```
            this.dbManager.executeSQLNonQuery(query);
        }
    }
}
```

ProjectRegistry/DataBase/Event/EventController.cs

```
using System;

namespace ProjectRegistry.DataBase.Event
{
    /// <summary>
    /// Handles data manipulation with Event table
    /// </summary>
    public class EventController : AbstractController
    {
        public EventController() : base()
        {
        }



        public void addEvent()
        {
            string query = @"INSERT INTO Events
                        (ProjectID, Name, Date, LinkToReport
)
                        VALUES
                        (@ProjectID, @Name, @Date, @LinkToRe
port)";
            this.dbManager.executeSQLNonQuery(query);
        }

        public void updateEvent(string eventID)
        {
            this.addParam("EventID",eventID);
            string query = @"UPDATE Events SET
                    ProjectID = @ProjectID, Name = @Name, Da
te = @Date,
                    LinkToReport = @LinkToReport
```

```
                WHERE Events.ID = @EventID";

            this.dbManager.executeSQLNonQuery(query);
        }



        public void deleteEvent(string id)
        {
            string query=@"UPDATE Events
                        SET ";
            query+=" Deleted= 1 ";
            query+=" Where Events.ID = '" + id + " '";
            this.dbManager.executeSQLNonQuery(query);
        }



    }
}
```

ProjectRegistry/DataBase/Project/ProjectController.cs

```
using System;
using System.Data;

namespace ProjectRegistry.DataBase.Project
{
    /// <summary>
    /// Handles data manipulation with Project table
    /// </summary>
    public class ProjectController : AbstractController
    {
        public ProjectController() : base() {}

        public void addProject()
        {
            string query=@"INSERT INTO Projects
                        (Code, Name, BeginDate, EndDate,Fini
shed, Description,
                        ProjectManagerID, ProjectTeam, PPK,
ProjectSize, Client,
                        ClientRepresentative, Contract, Proj
ectPlan, Deleted)
                        VALUES
                        (@Code,@Name,@BeginDate,@EndDate, @F
inished, @Description,
                        @ProjectManagerID,@ProjectTeam, @PPK
, @ProjectSize, @Client,
                        @ClientRepresentative, @Contract, @P
rojectPlan, 0)";
            this.dbManager.executeSQLNonQuery(query);
        }

        public void updateProject(string projectID)
```

```
        {
            this.addParam("ProjectID",projectID);
            string query=@"UPDATE Projects SET
                Code=@Code, Name=@Name, BeginDate=@BeginDate
, EndDate=@EndDate,
                Finished= @Finished, Description=@Descriptio
n,
                ProjectManagerID=@ProjectManagerID, ProjectT
eam=@ProjectTeam,
                PPK=@PPK, ProjectSize=@ProjectSize, Client=@
Client,
                ClientRepresentative=@ClientRepresentative,
Contract=@Contract,
                ProjectPlan=@ProjectPlan WHERE Projects.ID=@
ProjectID";

            this.dbManager.executeSQLNonQuery(query);
        }

        public void deleteProject(string id)
        {
            string query=@"UPDATE Projects
                        SET ";
            query+=" Deleted= 1 ";
            query+=" Where Projects.ID = '" + id + " '";
            this.dbManager.executeSQLNonQuery(query);
        }

    }
}
```

ProjectRegistry/DataBase/Task/TaskController.cs

```
using System;

namespace ProjectRegistry.DataBase.Task
{
    /// <summary>
    /// Handles data manipulation with Tasks table
    /// </summary>
    public class TaskController :AbstractController
    {
        public TaskController(): base() {}

        public void insertTask()
        {
            string query = @"INSERT INTO Tasks
                        (ProjectID, EmployeeID, Name, Planne
dHours, StartDate,
                        EndDate, Priority, Description, Acce
ptanceCriteria)
                        VALUES
                        (@ProjectID, @EmployeeID, @Name, @Pl
annedHours,
                        @StartDate, @EndDate, @Priority, @De
scription,
                        @AcceptanceCriteria)";
            this.dbManager.executeSQLNonQuery(query);
        }

        public void updateTask(string taskID)
        {
            this.addParam("TaskID",taskID);
            string query=@"UPDATE Tasks SET
                        ProjectID=@ProjectID, EmployeeID=@Em
```

```
ployeeID, Name=@Name,
                        PlannedHours=@PlannedHours, StartDat
e=@StartDate,
                        EndDate=@EndDate, ApprovedPM=@Approv
edPM, ApprovedDM=@ApprovedDM,
                        FinishDate=@FinishDate, UsedHours=@U
sedHours,
                        Priority=@Priority, Description=@Des
cription,
                        AcceptanceCriteria=@AcceptanceCriter
ia WHERE Tasks.ID=@TaskID";
            this.dbManager.executeSQLNonQuery(query);
        }

        public void deleteTask(string tid)
        {
            string query=@"UPDATE Tasks
                    SET ";
            query += " Deleted= 1 ";
            query += " Where Tasks.ID = '" + tid + " '";
            this.dbManager.executeSQLNonQuery(query);
        }
    }
}
```

subsectionDatabase Fillers

ProjectRegistry/DataBase/DataSets/AbstractDSFiller.cs

```
using System;
using System.Data;

namespace ProjectRegistry.DataBase.DataSets
{
    /// <summary>
    /// Filler classes fills data from DB tables. Each fille
r fills a specific set of data.
    ///
    /// </summary>
    public abstract class AbstractDSFiller
    {
        protected DataBase.DBManager dbManager;
        protected DataSet dataSet;

        public AbstractDSFiller(DataSet dataSet)
        {
            this.dbManager = new DBManager();
            this.dataSet = dataSet;
        }

        /// <summary>
        /// Given a column name in DB table, returns a colum
n id
        /// </summary>
        /// <param name="columnName"> column name</param>
        /// <returns> column in</returns>
        protected int getCID(string columnName)
        {
            int cid = this.dataSet.Tables[0].Columns.IndexOf
(columnName);
```

```
            if (cid == -1) throw new System.Exception("bad c
olumn name");
            return cid;
        }

        /// <summary>
        /// gets a Value from DS given a column name. Return
s value of the first row in DS
        /// </summary>
        public string getFromDS(string columnName)
        {
            System.Data.DataSet ds=this.dataSet;
            if (ds.Tables[0].Columns.Contains(columnName) ==
 false)
                throw new System.Exception("bad column name"
);
            int cid = ds.Tables[0].Columns.IndexOf(columnNam
e);
            return ds.Tables[0].Rows[0].ItemArray.GetValue(c
id).ToString();

        }
    }
}
```

ProjectRegistry/DataBase/DataSets/Departments/AllDepartment.xsd
ProjectRegistry/DataBase/DataSets/Departments/AllDepartment.xsx
ProjectRegistry/DataBase/DataSets/Departments/AllDepartment.cs
ProjectRegistry/DataBase/DataSets/Departments/AllDepartmentDSF.cs

```
using System;

namespace ProjectRegistry.DataBase.DataSets.Departments
{
    /// <summary>
    /// Fills a dataset with each attribute of Department ta
ble
    /// </summary>
    public class AllDepartmentDSF : ProjectRegistry.DataBase
.DataSets.AbstractDSFiller
    {
        public AllDepartmentDSF(System.Data.DataSet dataSet)
 : base(dataSet) {}

        /// <summary>
        /// Fills DS with all rows
        /// </summary>
        public void fillDataSet()
        {
            string select =
                @"SELECT Departments.ID as ID, Departments.N
ame as Name
                FROM Departments";

            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = select;
            this.dbManager.sqlAdapter.Fill(dataSet, "Departm
ents");
        }
```

```
        /// <summary>
        /// Fills DS with only one department
        /// </summary>
        public void fillDataSet(string id)
        {
            string select =
                @"SELECT Departments.ID as ID, Departments.N
ame as Name
                FROM Departments Where Departments.ID = '" +
 id + "'";

            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = select;
            this.dbManager.sqlAdapter.Fill(dataSet, "Departm
ents");
        }

        /// <summary>
        /// Makes an empty DS with one row, which has -1 or
"" values (used in Insert pages)
        /// </summary>
        public void fillDataSetEmpty()
        {
            string[] row = new string[this.dataSet.Tables[0]
.Columns.Count];
            row[this.getCID("ID")] = "-1";
            row[this.getCID("Name")] = "true";
            this.dataSet.Tables[0].Rows.Add(row);
        }

    }
}
```

ProjectRegistry/DataBase/DataSets/Employees/DSEmployees.xsd.xsd
ProjectRegistry/DataBase/DataSets/Employees/DSEmployees.xsd.xsx
ProjectRegistry/DataBase/DataSets/Employees/DSEmployees.xsd.cs
ProjectRegistry/DataBase/DataSets/Employees/EmployeesDSF.cs

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ProjectRegistry.DataBase.DataSets.Employees
{
    /// <summary>
    /// Fills DS with data from Employees table
    /// </summary>
    public class EmployeesDSF : ProjectRegistry.DataBase.Dat
aSets.AbstractDSFiller
    {

        public EmployeesDSF(DataSet dataSet) : base(dataSet)
 {}

        string query =
            @"SELECT Employees.ID,
                    Employees.Name AS Name,
                    Employees.Email, Employees.Working, Empl
oyees.WindowsLogIn,
                    Employees.Administrator,
                    Departments.Name AS DepartmentName,
                    Employees.DepartmentID as DepartmentID
                FROM Employees
                INNER JOIN
                    Departments ON Employees.DepartmentID =
Departments.ID";
```

```
        /// <summary>
        /// fills data set with all rows
        /// </summary>
        public void fillDataSet()
        {
            fillFilteredDateSet("");
        }

        protected void fillFilteredDateSet(string filter)
        {
            string query = this.query;
            if (filter != "")
                query += " WHERE " + filter;

            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = query;
            this.dbManager.sqlAdapter.Fill(dataSet, "Employe
es");
        }

        /// <summary>
        /// fills data set with only one row
        /// </summary>
        /// <param name="id"> id of the employee</param>
        public void fillDataSet(string id)
        {
            fillFilteredDateSet("Employees.ID = '" + id + "'
");
        }

        public void fillDataSetEmpty()
        {
```

```
            string[] row = new string[this.dataSet.Tables[0]
.Columns.Count];
            row[this.getCID("ID")] = "-1";
            row[this.getCID("Working")] = "True";
            row[this.getCID("DepartmentID")] = "-1";
            row[this.getCID("DepartmentName")] = "";
            row[this.getCID("Administrator")] = "False";
            this.dataSet.Tables[0].Rows.Add(row);
        }

    }
}
```

ProjectRegistry/DataBase/DataSets/Events/DSEvents.xsd.xsd ProjectRegistry/DataBase/DataSets/Events/DSEvents.xsd.xsx ProjectRegistry/DataBase/DataSets/Events/DSEvents.xsd.cs

ProjectRegistry/DataBase/DataSets/Events/EventsDSF.cs

```
using System;
using System.Data;

namespace ProjectRegistry.DataBase.DataSets.Events
{
    /// <summary>
    /// Fills DS with data from Events table
    /// </summary>
    public class BatchEventsDSF : ProjectRegistry.DataBase.D
ataSets.AbstractDSFiller
    {

        public BatchEventsDSF(DataSet dataSet) : base(dataSe
t) {}

        string selectEvents = @"SELECT Events.ID as ID,
                Events.ProjectID as ProjectID, Events.Name a
s Name,
                Events.Date as Date, Events.LinkToReport as
LinkToReport,
                Projects.Code as ProjectCode
                FROM Events
                INNER JOIN Projects ON Projects.ID = Events.
ProjectID";

        /// <summary>
        /// fills DS will all rows from table
        /// </summary>
        public void fillDataSet()
```

```
        {
            string query = selectEvents;

            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = query;
            this.dbManager.sqlAdapter.Fill(dataSet, "Events"
);

        }

        public void fillDataSet(string eventID)
        {
            string query = selectEvents + " Where Events.Id
= '" + eventID + "'";
            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = query;
            this.dbManager.sqlAdapter.Fill(dataSet, "Events"
);
        }

        public void fillDataSetEmpty()
        {
            string[] row = new string[this.dataSet.Tables[0]
.Columns.Count];
            row[this.getCID("ID")] = "-1";
            row[this.getCID("Name")] = "True";
            row[this.getCID("ProjectID")] = "-1";
            row[this.getCID("Date")] = "";
            row[this.getCID("LinkToReport")] = "False";
            this.dataSet.Tables[0].Rows.Add(row);
        }
    }
```

```
}
```

ProjectRegistry/DataBase/DataSets/Years/DSYears.xsd ProjectRegistry/DataBase/DataSets/Years/DSYears.xsx ProjectRegistry/DataBase/DataSets/Years/DS

ProjectRegistry/DataBase/DataSets/Years/YearsDSF.cs

```
using System;
using System.Data;

namespace ProjectRegistry.DataBase.DataSets.Years
{
    /// <summary>
    /// builds a DS, which shows year numbers.
    /// </summary>
    public class YearsDSF : ProjectRegistry.DataBase.DataSet
s.AbstractDSFiller
    {

        public YearsDSF(DataSet dataSet) : base(dataSet) {}

        public void fillDataSet()
        {
            for (int i = 1998; i <= DateTime.Now.Year; i++)

            {
                object[] row = new object[1];
                row[0] = i;
                dataSet.Tables[0].Rows.Add(row);
            }
        }
    }
}
```

ProjectRegistry/DataBase/DataSets/ProjectManagers/PMs.xsd ProjectRegistry/DataBase/DataSets/ProjectManagers/PMs.xsx ProjectRegistry/DataBase/DataSets/ProjectManagers/PMs.cs

ProjectRegistry/DataBase/DataSets/ProjectManagers/PMsDSF.cs

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ProjectRegistry.DataBase.DataSets.ProjectManagers
{
    /// <summary>
    /// Fills data set with info about Project Managers
    /// </summary>
    public class PMsDSF : AbstractDSFiller
    {

        public PMsDSF(DataSet dataSet) : base(dataSet) {}

        public void fillDataSet()
        {

            string selectPMs = @"SELECT DISTINCT Employees.I
D, Employees.Name
                FROM Projects
                INNER JOIN Employees ON Projects.ProjectMana
gerID = Employees.ID
                WHERE Projects.deleted = 0";

            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = selectPMs;
            this.dbManager.sqlAdapter.Fill(dataSet, "PManage
rs");
        }
```

```
    }
}
```

ProjectRegistry/DataBase/DataSets/Projects/AllProject.xsd ProjectRegistry/DataBase/DataSets/Projects/AllProject.xsx ProjectRegistry/DataBase/DataSets/Projects/AllProject.cs

ProjectRegistry/DataBase/DataSets/Projects/AllProjectDSF.cs

```
using System;

namespace ProjectRegistry.DataBase.DataSets.Projects
{
    /// <summary>m
    /// Fills DS with all data from Projects table
    /// </summary>
    public class AllProjectDSF:ProjectRegistry.DataBase.Data
Sets.AbstractDSFiller
    {
        public AllProjectDSF(System.Data.DataSet dataSet) :
base(dataSet) {}

        public void fillDataSet(string projectID)
        {
            string selectProjects =  @"SELECT
                Projects.ID, Projects.Code, Projects.Name, P
rojects.BeginDate,
                Projects.EndDate, Projects.Finished, Project
s.Description,
                Projects.ProjectManagerID as PMID, Projects.
ProjectTeam, Projects.PPK,
                Projects.ProjectSize, Projects.Client, Proje
cts.ClientRepresentative,
                Projects.Contract, Projects.ProjectPlan, Pro
jects.Deleted,
                Employees.Name as PMName
                FROM Projects
                INNER JOIN Employees
```

```
                    ON Projects.ProjectManagerID = Employees
.ID
                WHERE
                    Projects.ID= '"+projectID +"'";
            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = selectProjects;
            this.dbManager.sqlAdapter.Fill(dataSet, "Project
s");
        }
    }
}
```

ProjectRegistry/DataBase/DataSets/Projects/DSProjectCode.xsd.xsd
ProjectRegistry/DataBase/DataSets/Projects/DSProjectCode.xsd.xsx
ProjectRegistry/DataBase/DataSets/Projects/DSProjectCode.xsd.cs
ProjectRegistry/DataBase/DataSets/Projects/ProjectCodeDSF.cs

```
using System;

namespace ProjectRegistry.DataBase.DataSets.Projects
{
    /// <summary>
    /// Fills data with Project.ID and Project.Code
    /// </summary>
    public class ProjectCodeDSF : ProjectRegistry.DataBase.D
ataSets.AbstractDSFiller
    {

        public ProjectCodeDSF(System.Data.DataSet dataSet) :
 base(dataSet) {}

        string query = @"SELECT
                    Projects.ID, Projects.Code
                FROM Projects
                    Where Projects.Deleted = 0";



        public void fillDataSet()
        {
            string selectProjects = query;

            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = selectProjects;
            this.dbManager.sqlAdapter.Fill(dataSet, "Project
s");
        }
```

```
        public void fillDataSet(string projectID)
        {
            string selectProjects = query + "WHERE Projects.
ID= '" + projectID + "'";
            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = selectProjects;
            this.dbManager.sqlAdapter.Fill(dataSet, "Project
s");
        }



    }
}
```

ProjectRegistry/DataBase/DataSets/Projects/DSProjects.xsd ProjectRegistry/DataBase/DataSets/Projects/DSProjects.xsx ProjectRegistry/DataBase/DataSets/Projects/DSProjects.cs

ProjectRegistry/DataBase/DataSets/Projects/ProjectsDSF.cs

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ProjectRegistry.DataBase.DataSets.Projects
{
    /// <summary>
    /// Creates DS with data from projects table
    /// </summary>
    public class ProjectsDSF : ProjectRegistry.DataBase.Data
Sets.AbstractDSFiller
    {

        public ProjectsDSF(DataSet dataSet) : base(dataSet)
{}

        private string selectProjects =
            @"SELECT
                    Projects.ID, Projects.Code, Projects.Nam
e,
                    Projects.Code + ':' + Projects.Name as E
xtendedName,
                    Projects.BeginDate, Projects.EndDate,
                    Projects.Finished,
                    Employees.Name AS PMName,
                    Projects.Client,
                    Projects.ProjectManagerID AS PMId
                FROM Projects
                INNER JOIN Employees
```

```
                    ON Projects.ProjectManagerID = Employees
.ID";

        public void fillDataSet()
        {
            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = selectProjects;
            this.dbManager.sqlAdapter.Fill(dataSet, "Project
s");
        }

        public void fillDataSet(string projectID)
        {
            string sp = selectProjects + " WHERE Projects.ID
= '" + projectID + "'";
            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = sp;
            this.dbManager.sqlAdapter.Fill(dataSet, "Project
s");
        }



    }
}
```

ProjectRegistry/DataBase/DataSets/Projects/ProjectStatusDSF.cs

```
using System;
using System.Data;

namespace ProjectRegistry.DataBase.DataSets.Projects
{
    /// <summary>
    /// creates a DS with two values:  Uzdarytas and Aktyvus

    /// </summary>
    public class ProjectStatusDSF : ProjectRegistry.DataBase
.DataSets.TrueFalse.TrueFalseDSF
    {
        public ProjectStatusDSF(DataSet dataSet) : base(data
Set, "Uzdarytas", "Aktyvus") {}
    }
}
```

ProjectRegistry/DataBase/DataSets/Tasks/DSTasks.xsd ProjectRegistry/DataBase/DataSets/Tasks/DSTasks.xsx ProjectRegistry/DataBase/DataSets/Tasks/DS

ProjectRegistry/DataBase/DataSets/Tasks/TasksDSF.cs

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ProjectRegistry.DataBase.DataSets.Tasks
{
    /// <summary>
    /// Fills DS with data from Tasks table
    /// </summary>
    public class TasksDSF : ProjectRegistry.DataBase.DataSet
s.AbstractDSFiller
    {

        public TasksDSF(DataSet dataSet) : base(dataSet) {}

        private string selectTasks =
            @"SELECT Tasks.ID as ID, Projects.Name AS Projec
tName,
                    Employees.Name AS EmployeeName,
                    Tasks.Name AS TaskName, Tasks.PlannedHou
rs,
                    Tasks.StartDate, Tasks.EndDate,
                    Tasks.ApprovedPM, Tasks.ApprovedDM, Task
s.FinishDate,
                    Tasks.UsedHours, Tasks.ProjectID as Proj
ectID,
                    Tasks.EmployeeID, Projects.Code as Proje
ctCode,
                    Tasks.Description as Description
                FROM Tasks
```

```
                INNER JOIN
                    Projects ON Tasks.ProjectID = Projects.I
D
                INNER JOIN
                    Employees ON Tasks.EmployeeID = Employee
s.ID";

        public void fillDataSet()
        {
            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = selectTasks;
            this.dbManager.sqlAdapter.Fill(dataSet, "Tasks")
;
        }

        public void fillDataSet(string taskID)
        {
            string st = selectTasks + " Where Tasks.ID = '"
+ taskID + "'";
            this.dbManager.sqlAdapter.SelectCommand.CommandT
ext = st;
            this.dbManager.sqlAdapter.Fill(dataSet, "Tasks")
;
        }

    }
}
```

ProjectRegistry/DataBase/DataSets/TrueFalse/DSTrueFalse.xsd ProjectRegistry/DataBase/DataSets/TrueFalse/DSTrueFalse.xsx ProjectRegistry/DataBase/DataSets/TrueFalse/DSTrueFalse.cs

ProjectRegistry/DataBase/DataSets/TrueFalse/TrueFalseDSF.cs

```
using System;
using System.Data;

namespace ProjectRegistry.DataBase.DataSets.TrueFalse
{
    /// <summary>
    /// Summary description for ProjectStatusDSF.
    /// </summary>
    public class TrueFalseDSF : ProjectRegistry.DataBase.Dat
aSets.AbstractDSFiller
    {

        private string True;
        private string False;

        public TrueFalseDSF(DataSet dataSet) : this(dataSet,
 "Taip", "Ne") {}

        public TrueFalseDSF(DataSet dataSet, string True, st
ring False) : base(dataSet)
        {
            this.True = True;
            this.False = False;
        }

        public void fillDataSet()
        {
            object[] row = new object[2];
            row[0] = false; row[1] = False;
```

```
            dataSet.Tables[0].Rows.Add(row);

            row = new object[2];
            row[0] = true; row[1] = True;
            dataSet.Tables[0].Rows.Add(row);
        }
    }
}
```

T.3 Common Library

PRCommonLibrary/Persistency/FilterHashTable.cs

```
using System;

namespace PRCommonLibrary.Persistency
{



    public class FilterHashTable: SessionPersistency
    {
        /// <summary>
        /// Defines filtering rules.
        ///
        /// Each filter rule has at least one value. The sec
ond value is optional, as only
        /// some rules utilizes it (e.g. InRange).
        /// </summary>
        /// <remarks> <see cref="FilterHashTable.buildFilter
"/> </remarks>
        public class FilterRule
        {
            public enum FilterOperation {NoFilter, Equals, L
essOrEqual,
                EqualOrHigher, InRange, IsLike, InYear, Sort
ing};

            public FilterOperation fOperation;
            public string value1;
            public string value2;

            public FilterRule(FilterOperation op, string val
ue1, string value2)
```

```
            {
                this.fOperation = op;
                this.value1 = value1;
                this.value2 = value2;
            }
        }

        /// <summary>
        /// Creates a FilterHash Table
        /// </summary>
        /// <param name="id"> an id, which uniquely identifi
es this object between other SessionPersistency objects </pa
ram>
        public FilterHashTable(string id) : base(id)
        {
        }

        protected override object createNewObject()
        {
            return new System.Collections.Hashtable();
        }

        protected System.Collections.Hashtable getHashTable(
)
        {
            return this.getPersistenObject() as System.Colle
ctions.Hashtable;
        }

        /// <summary>
        /// Adds a filter rule, and deletes an existing one,
 if any. if rule is "-1",
        /// deletes filter
```

```
        /// </summary>
        /// <param name = "filter"> a filter string. For exa
mple a collumn name for a drop down list </param>
        /// <param name = "rule"> a filter value </param>
        /// <example> <code> BatchView.addFilterRuleSingle("
ProjectCode", "AXCF") </code> </example>

        public void injectRule(string columnName, FilterRule
.FilterOperation operation,
            string colValue1, string colValue2)
        {
            this.removeRule(columnName);
            if (operation == FilterRule.FilterOperation.NoFi
lter) return;
            if ((colValue1 == "-1") || (colValue1 == ""))ret
urn;
            //if ((colValue2 == "-1") || (colValue2 == ""))
return;
            this.getHashTable().Add(columnName, new FilterRu
le(operation, colValue1, colValue2));
        }

        public void removeRule(string columnName)
        {
            if (getHashTable().ContainsKey(columnName))
                getHashTable().Remove(columnName);
        }

        public FilterRule getFilterRule(string columnName)
        {
            if (this.getHashTable()[columnName] == null) ret
urn null;
            return this.getHashTable()[columnName] as Filter
```

```
Rule;
        }



        /// <summary>
        /// Removes all rules
        /// </summary>
        public void removeRules()
        {
            this.getHashTable().Clear();
        }



        /// <summary>
        /// Builds a string combining all filter rules. exam
ple result: "Code = 'axa' AND ID=1"
        ///
        /// </summary>
        /// <returns></returns>
        public string buildFilter()
        {
            bool bAddAnd = false;
            string rule = "";
            System.Collections.IDictionaryEnumerator e = thi
s.getHashTable().GetEnumerator();
            while (e.MoveNext())
            {
                if (bAddAnd)
                    rule += " AND ";
                else
                    bAddAnd = true;
                FilterRule fr = e.Value as FilterRule;
                string eval = "";
```

```
                if (fr.fOperation == FilterRule.FilterOperat
ion.Equals)
                    eval = "{0} = {1}"; else
                if (fr.fOperation == FilterRule.FilterOperat
ion.IsLike)
                    eval = "{0} LIKE '*{1}*'"; else
                if (fr.fOperation == FilterRule.FilterOperat
ion.InYear)
                    eval = "{0} > '01/01/{1}' AND {0} < '12/
31/{1}'"; else
                if (fr.fOperation == FilterRule.FilterOperat
ion.Sorting)
                    eval = "{0} {1}";
                rule += String.Format(eval, e.Key.ToString()
, fr.value1, fr.value2);
            }
            return rule;
        }

    }

}
```

PRCommonLibrary/Persistency/SessionPersistency.cs

```
using System;

namespace PRCommonLibrary.Persistency
{
    /// <summary>
    /// Makes a class persistent, by saving it into session
    /// </summary>
    abstract public class SessionPersistency
    {
        private string id;
        private System.Web.SessionState.HttpSessionState ses
sion;

        public SessionPersistency(string id)
        {
            this.id = id;
            this.session = null;
        }

        public string PageID
        {
            get{ return this.id;}
        }

        public System.Web.SessionState.HttpSessionState Page
Session
        {
            get
            {
                if (this.session == null) throw new System.E
xception("must set PageSession before use");
                return this.session;
```

```
            }
            set {this.session = value;}
        }



        /// <summary>
        /// Creates an object, which will be saved in a sess
ion
        /// </summary>
        /// <returns> a new object</returns>
        protected abstract object createNewObject();

        protected object getPersistenObject()
        {
            if (session[id] == null)
            {
                session[id] = this.createNewObject();
            }
            return session[id];
        }



    }

}
```

T.4 Custom Controls

PRControls/DatePicker.cs

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;

namespace PRControls
{
    /// <summary>
    /// A calendar
    /// </summary>
    [DefaultProperty("Text"),
        ToolboxData("<{0}:DatePicker runat=server></{0}:Date
Picker>")]
    public class DatePicker : System.Web.UI.WebControls.Text
Box
    {

        public DatePicker(): base()
        {
            Attributes["elemDateInputs"] = "this";
            Attributes["popupType"] = "both";
            Attributes["pickerDateFormat"] = "[MMMM] [yyyy]"
;
            Attributes["inputDateFormat"] = "[yyyy]-[MM]-[dd
]";
            Attributes["firstDayOfWeek"] = "1";
            Attributes["daynameLetters"] = "S;P;A;T;K;P;S;";

            Attributes["monthnamesLong"] = "Sausis;Vasaris;K
ovas;Balandis;Geguze;Birzelis;Liepa;Rugpjutis;Rugsejis;Spali
```

```
s;Lapkritis;Gruodis;";
            Attributes["initialEmpty"] = "1";
        }

        /// <summary>
        /// Render this control to the output parameter spec
ified.
        /// </summary>
        /// <param name="output"> The HTML writer to write o
ut to </param>
        protected override void Render(HtmlTextWriter output
)
        {
            string buttonID = "btn" + UniqueID;
            Attributes["elemDateButtons"] = buttonID;

            base.Render(output);
            output.Write("<BUTTON id=\"" + buttonID + "\" ty
pe=\"button\" class=\"" + CssClass + "\" />");
        }
    }
}
```

PRControls/FilterDropDownList.cs

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using PRCommonLibrary;

namespace PRControls
{
    /// <summary>
    ///   PRDropDownList extends System.Web.UI.WebControls.D
ropDownList and
    /// adds such new features:
    /// 1. You can add an extra item which will be as a head
er. E.g. "Select...",
    /// 2. You can specify a Value of an item which will be
selected by default.
    /// 3. You can specify a FilterOperation
    /// </summary>
    [DefaultProperty("DefaultValueText"),
    ToolboxData("<{0}:FilterDropDownList runat=server></{0}:
FilterDropDownList>")]
    public class FilterDropDownList : System.Web.UI.WebContr
ols.DropDownList, System.Web.UI.INamingContainer
    {
        private string defaultValueText;
        private string defaultValueID;
        private string filterColumnName;
        private PRCommonLibrary.Persistency.FilterHashTable.
FilterRule.FilterOperation filterOperation;

        /// <summary>
        /// Default Value
```

```
        /// </summary>
        [Bindable(true), Category("Appearance"), DefaultValu
e("<nefiltruoti>")]
        public string DefaultValueText
        {
            get
            {
                if (defaultValueText != null)
                    return defaultValueText;
                else
                    return "<nefiltruoti>";
            }

            set
            {
                defaultValueText = value;
            }
        }

        /// <summary>
        /// Default Value ID
        /// </summary>
        [Bindable(true), Category("Appearance"), DefaultValu
e("-1")]
        public string DefaultValueID
        {
            get
            {
                if (defaultValueID != null)
                    return defaultValueID;
                else
                    return "-1";
            }
```

```
            set
            {
                defaultValueID = value;
            }
        }

        /// <summary>
        /// Filter Column Name
        /// </summary>
        [Bindable(true), Category("Appearance"), DefaultValu
e("")]
        public string FilterColumnName
        {
            get
            {
                return filterColumnName;
            }

            set
            {
                filterColumnName = value;
            }
        }



        /// <summary>
        /// Filter Operation
        /// </summary>
        public PRCommonLibrary.Persistency.FilterHashTable.F
ilterRule.FilterOperation FilterOperation
        {
```

```
            get
            {
                return this.filterOperation;
            }
            set
            {
                this.filterOperation = value;
            }
        }

        /// <summary>
        /// adds a default item to Filter DDL
        /// </summary>
        protected override void OnDataBinding(EventArgs e)
        {
            base.OnDataBinding(e);
            if (DefaultValueText != "")
                Items.Insert(0, new ListItem(DefaultValueTex
t, DefaultValueID));
        }

    }

}
```

PRControls/FilterTextBox.cs

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using PRCommonLibrary;

namespace PRControls
{
    /// <summary>
    /// A custom text box, which has a FilterOperation and F
ilter Column values
    /// </summary>
    [DefaultProperty("Text"),
        ToolboxData("<{0}:FilterTextBox runat=server></{0}:F
ilterTextBox>")]
    public class FilterTextBox : System.Web.UI.WebControls.T
extBox
    {
        private string filterColumnName;
        private PRCommonLibrary.Persistency.FilterHashTable.
FilterRule.FilterOperation filterOperation;

        /// <summary>
        /// FilterColumnName
        /// </summary>
        [Bindable(true), Category("Appearance"), DefaultValu
e("")]
        public string FilterColumnName
        {
            get
            {
                return filterColumnName;
```

```
            }

            set
            {
                filterColumnName = value;
            }
        }

        /// <summary>
        /// FilterOperation
        /// </summary>
        public PRCommonLibrary.Persistency.FilterHashTable.F
ilterRule.FilterOperation FilterOperation
        {

            get
            {
                return this.filterOperation;
            }
            set
            {
                this.filterOperation=value;
            }
        }
    }
}
```

T.5 Tests

PRTests/PageExistenceTests.cs

```
using System;
using NUnit.Extensions.Asp;
using NUnit.Extensions.Asp.AspTester;

namespace PRTests
{
    /// <summary>
    /// Summary description for Class1.
    /// </summary>
    public class PageExistenceTests : WebFormTestCase
    {

        public void TestBatchProjectViewPageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/BatchViews/BatchProjectView.aspx");
        }

        public void TestBatchTaskViewPageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/BatchViews/BatchTaskView.aspx");
        }

        public void TestBatchEmployeeViewPageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/BatchViews/BatchEmployeeView.aspx");
        }
```

```
        public void TestBatchEventViewPageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/BatchViews/BatchEventView.aspx");
        }

        public void TestAdministratePageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/Administrator/Administrate.aspx");
        }

        public void TestViewProjectPageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/DetailView/ViewProject.aspx?ProjectID=1");
        }

        public void TestInsertEmployeePageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/Inserts/InsertEmployee.aspx");
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/Inserts/InsertEmployee.aspx?EmployeeID=1");
        }

        public void TestInsertProjectPageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/Inserts/InsertProject.aspx");
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/Inserts/InsertProject.aspx?ProjectID=1");
        }
```

```
        public void TestInsertTaskPageExistence()
        {
            Browser.GetPage("http://localhost/ProjectRegistr
y/Pages/Inserts/InsertTask.aspx");
        }

    }
}
```

This Document was prepared by:

Vilma Rudžionytė: _____

Darius Damalakas: _____

Martynas Kriaučiūnas: _____

Viktoras Čiumanovas: _____

Date and place 07-11-2005 Roskilde