CASE STUDY FOR A LIGHTWEIGHT IMPACT ANALYSIS TOOL

A thesis submitted
to Kent State University in partial fulfillment of the requirements for the degree of Master’s of Science

by
Alice Lewis
May, 2009
Thesis written by
Alice Lewis
B.A., Hiram College, Hiram, Ohio, USA 1974
M.A., Kent State University, Kent, Ohio, USA, 1977
M.A., Loyola University, New Orleans, LA, USA, 1997
M.S., Kent State University, Kent, Ohio, USA, 2009

Approved by
_______________________ Jonathan Maletic_, Advisor
_______________________ Robert Walker_____, Chair, Department of Computer Science
_______________________ John Stalvey_____, Dean, College of Arts and Sciences
TABLE OF CONTENTS

DEDICATION .................................................................................................................. V

ACKNOWLEDGEMENTS ........................................................................................... VI

CHAPTER 1 INTRODUCTION .................................................................................. 1

1.1.1 Motivation ....................................................................................................... 2

1.2 MinImpact ............................................................................................................. 5

1.3 Thesis Organization .............................................................................................. 6

CHAPTER 2 RELATED WORK ............................................................................... 7

CHAPTER 3 THE MINIMPACT APPROACH ....................................................... 9

3.1 Tool requirements ................................................................................................ 9

3.2 Development of MinImpact ................................................................................. 11

3.2.1 Legal considerations ...................................................................................... 12

3.2.2 Gathering and delivering the code ................................................................. 12

3.2.3 System Development ...................................................................................... 12

3.2.4 Design and Evaluation .................................................................................... 16

3.2.5 Interface ......................................................................................................... 19

3.3 Implementation details ........................................................................................ 19

CHAPTER 4 CASE STUDY .................................................................................... 22

4.1 Feedback on Use of MinImpact ........................................................................... 22

4.2 Changes in this release: ...................................................................................... 23

4.3 Case Study Findings ............................................................................................ 24

CHAPTER 5 SUMMARY AND CONCLUSIONS ............................................. 25
DEDICATION

This thesis is dedicated to the author’s family without whose support this would not have been possible.
ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Dr. Maletic, and Tim Fontaine, the undergraduate student who wrote MinImpact.

Alice Lewis

March 31, 2009 Defense Date, Kent, Ohio
CHAPTER 1

Introduction

Impact Analysis is defined in the IEEE Standard for Software Maintenance (various 2006) as “impact analysis: Identifies all system and software products that a change request affects and develops an estimate of the resources needed to accomplish the change. This includes determining the scope of the changes to plan and implement work, accurately estimating the resources needed to perform the work, and analyzing the requested change’s cost and benefits.”

Impact analysis is a necessary part of performing high quality software maintenance, but is a very costly side effect of maintaining software. As a result, most development teams do not perform an impact analysis due to the costs. At best, only a cursory look at surrounding systems is done – after all, bug fixes and many enhancements take less than an hour to code – how can the team justify working hours on an impact analysis? At worst, there is no consideration of the impact of an upcoming bug fix; it is fixed in isolation and installed in production.

The maturity of Impact Analysis in practice is very low. Sometimes the entire burden is placed on the testing team to “test everything” for even the smallest code change. The result is introduction of new, sometimes serious defects into production. A low cost solution is needed that produces at least some reasonable results.
To address this problem we’ve created a lightweight impact analysis tool that interfaces with the development team’s configuration management system. The tool encompasses a simple, non-invasive way to determine components of the system impacted by the proposed change.

### 1.1.1 Motivation

As much as 90% (Seacord, Plakosh et al. 2003) of all software development is maintenance. This means that “new development” represents only 10% of the current software development effort taking place today. Maintenance is being accomplished on legacy systems that were originally developed prior to any “newer” software development methodologies. Documentation, if it ever existed, is out-of-date. The problem of determining how and what is impacted by a given change is called *Impact Analysis* (Arnold 1996).

Determining the impact of a code change is a persistent problem in any real world development. Doing a thorough job of impact analysis, even for a small change, to a major system is a daunting and costly task. This holds true even if all documentation and traceability among deliverables is available. In actual practice, documentation and traceability among them does not exist for most systems and most development teams do little in the way of impact analysis. As a result, many new defects are introduced due to bug fixes or feature additions.

Arnold (Arnold 1996) discusses different approaches to impact analysis and they are as varied as there are programmers who develop these increasingly complex systems. If the system was properly documented, the task of combing through all the old
documentation, including requirements, design, code, unit test procedures, system test procedures, installation procedures, etc. can make the cost of even a small code change difficult to justify. If the system is not documented, the task is even more monumental and may consist in part of creating that documentation.

One alternative to performing a full impact analysis is to fix a problem in a portion of code, test only that change, and release as is, hoping that the author has not impacted any other aspects of the system. However, this often leads to other defects. The tools that are available for performing impact analyses often dictate that the software must be initially developed using that tool, thereby requiring buy-in to a specific software process. In addition, those toolsets are typically very expensive.

A change request is usually requested via a bug tracking tool, assigned to a programmer. The programmer then codes it and releases it to test. The tester often reproduces the bug with old code, installs new code and assures that the bug is gone (or feature is changed, as applicable). When testers are not available, even this level of diligence is not performed. Because of tight schedules, there is at best only a cursory check for the impact of a change to a common unit – these changes are often made to fix an error in production that is negatively affecting the user community and the assigned goal is to fix one problem – not scope out the possible impact on other systems. In fact, without proof that the developer has impacted another system, he/she will not be allowed to perform the due diligence necessary.

The message from management is clearly to fix the problem as soon as possible. The developer often does just that, then crosses his/her fingers that there will be no
adverse side effects. Although there are repercussions to introducing new problems in production, they are not as severe as not acting immediately and “fixing” the original problem.

One approach to this problem is to query the entire development community for use of the code/data in question. Real examples of such email correspondence related to this issue are the following two snippets:

“Hey, guys and gals – I need to change the above mentioned common unit. I know it is used by modules x1, x2, and x3 and that they need to be recompiled and retested. Does anyone else use this common unit? Speak now or forever hold your peace, because this has to go to production by the end of the month.”

“Everybody – I need to make a change to the above stored procedure to fix a critical defect in production. Without it, our most important client, Genghis Khan, cannot review his monthly financial statement. This is going into production NOW so I hope no one else uses it because if they do, I am not sure what this change will do to their system. And NO – this is NOT an April Fool’s joke.”

These small changes are often made to remediate critical problems in production. By fixing isolated problems, others are often introduced in ancillary applications that the developers, in their haste, did not know had any relationship to the code in question.
We are attempting to alleviate the dilemma faced by every software developer who is asked to make a “small” change and spend a minimal amount of time on it, yet to produce a quality product that will not impact other elements of the same system. A tool, “MinImpact” was created to accept a search string and query the entire configuration management database for a match. Using this tool, the developer and his/her management can determine the risk of the change and the resultant work that would have to be performed.

Knowing that use of a new tool can be resisted by the development community, a basic tool was created at first, with improvements suggested and implemented during an initial pilot phase and as part of the execution of the project itself. By so doing, we hoped to create a feeling of ownership for the tool to encourage continued use.

1.2 MinImpact

The approach exploits the use of configuration management of all source code as a way to search for software components impacted by the proposed change. This includes one extra step for the developer whenever a unit or data element is changed. Because the developer cannot guarantee the item is not used by other applications, the step would scan all of the latest source code under software configuration management (SCM) for the string in question – not just the source code for his application.

Impact analysis is important to all software maintenance projects. However, it is quite costly to do in a heavy weight manner. Organizations don’t want to take on this cost. So as an alternative we propose the use of a lightweight approach based on
configuration management. The contribution of this thesis is to apply my lightweight approach in a case study to a real world environment and assess the results.

1.3 Thesis Organization

The thesis is organized as follows. Chapter 1 – Overview of the thesis and motivation for the case study. Chapter 2 – Related work in the areas of impact analysis and software configuration management. Chapter 3 – Approach, design, implementation, and development iterations of the tool MinImpact in its pilot phase prior to the execution of the case study. Chapter 4 – The case study itself, conducted using the MinImpact tool. This chapter includes introduction of the case study to the company, gathering buy-in from developers to participate, activities during the case study, and initial observations. Chapter 5 – Conclusions and further research. This thesis merely scratches the surface of what could be accomplished with MinImpact. This chapter explores what further steps can be taken.
CHAPTER 2

Related Work

The publication, “Automated Impact Analysis of UML Models” (Briand, Labiche et al. 2006) proposes use of existing UML designs to help perform impact analyses during the maintenance phase of a project. This is useful for projects that have used UML methodology throughout, but as pointed out in the introduction, it isn’t very helpful for systems developed without the UML methodology.

Likewise, the publication “Automated Impact Analysis of Object-Oriented Software Systems” (Hoffman 2003) similarly requires that the entire system was designed and created in an object-oriented manner.

A lightweight approach was proposed in (Walker, Holmes et al. 2006) but was too complex to be of use to a general development community.

Use of the SCM tool for other than conventional SCM has also been proposed in the literature. One example which exploits the use of SCM is “Fine Grained Indexing of Software Repositories to Support Impact Analysis” (Canfora and Cerulo 2006), but the proposal contains a burdensome front-end (to accomplish indexing of the entire system) which must be built into the system. Another example is “Maintenance with Reuse: An Integrated Approach Based on Software Configuration Management” (Kwon, Shin et al. 1999) which introduced a method for using SCM as the basis for decisions and implementations of reuse.
Kagdi (Kagdi, Hammad et al. 2008) proposed in “Who Can Help Me with this Source Code Change?” that SCM be exploited to determine who touched the target source code in order to identify the expert to consult when the next change is proposed. This thesis also proposes that SCM history be exploited, but using a different approach than the MinImpact® tool.

Marcus et al (Marcus 2003) discusses an information retrieval method called Latent Semantic Indexing which is used to locate features and concepts within the code. This related work examined an existing system for a feature using natural language. The retrieval method uses as input a feature and searches the code for existence of that feature. MinImpact uses code as input (the changed module, stored procedure or subroutine name) and produces code as output.
CHAPTER 3

The MinImpact Approach

In this chapter the MinImpact approach is described and the process to develop the tool is discussed. The thrust behind the MinImpact approach is to examine the code being changed through use of the MinImpact tool. This examination will point out other parts in the entire system that shares the same code. When the developer proposes a change to code, she enters into the tool the name of the module or component or call that is being changed. The code could be a large module or a small stored procedure. MinImpact accepts this phrase as input and searches for its existence throughout the universe of SourceSafe configuration management.

MinImpact returns a list of all other parts of the system that reference the phrase entered. The list represents the parts of the system which should be examined to assure that the changed code will not negatively impact those parts. The next step in the process is for the developer to examine the code in the list to determine if further steps need to be taken to assure that code is not compromised by the change being proposed.

3.1 Tool requirements

The following three areas of software are the common culprits in this organization when changes have unanticipated side effects, so we will concentrate on examining these:

- common unit/component (code)
- common stored procedure (code)
• common data element

Attached as Appendix C are the original requirements sent to the developer. Appendix D represents the Use Cases provided to the developer.

Because the one process that all seem to agree on is the standard use of configuration management in this company, MinImpact exploits that fact. Perhaps there is less emotion around it or perhaps there are more examples of “good” configuration management. Although configuration management can be a pervasive problem, it can often be implemented by one person who really cares about the process and follows through. The same cannot be said of the other phases listed above. Clearly, there are many different techniques and tools to achieve configuration management that have historically succeeded. Failure at this stage is easy to track – a configuration management problem is not often recorded as a defect in the code or the requirements; whereas failure at requirements is often misdiagnosed as a coding or testing failure. When configuration management fails, it is easier to pinpoint root cause and resolution – much easier than in the case of a design issue. Therefore, the requirements for the tool are based on the good practice of configuration management by the organization.

Rarely is configuration management viewed across projects in an organization. Typically, a configuration manager is not responsible for all applications, but only for those in which he/she is expert or actively involved in coding. In order for MinImpact to be successful, it must be invoked across all projects. This will mitigate the risk for projects that the developer is not familiar with and “assumes” are not involved with the common code or data element in question.
3.2 Development of MinImpact

We employed a Kent State University undergraduate student, Tim Fontaine, to assist in the development of the MinImpact tool. It is implemented as a wrapper on top of SourceSafe (by Microsoft). A user friendly front end with screen and hard copy output allowed the organization to help assess the risk and estimate the subsequent work to assure that “small” changes do not negatively impact other applications. The tool was used by a set of volunteers for a period of time to measure its effectiveness. The high level requirements for the tool follow.

MinImpact® will accept as input the following:

a. The name of the item in question that is inserted in the “code” or “data element” prompt (see figure 1)
b. The population of code to be searched, including possible date parameters
c. All code that references the code segment/data element input along with its associated application
d. One page output per application
e. Notation of when the referenced code was last modified and by whom (see figure 2)

Requirements were produced and discussed with the student in a meeting with the author, Dr. Maletic, and Tim Fontaine. Tim asked for sample code so that he could debug and test the MinImpact product.
3.2.1 Legal considerations

Due to the fact that the programmer would be seeing company source code, NDA (non-disclosure agreement) considerations had to be handled. In order to avoid developing a NDA, we included only every fifth line of code from the source code population. This satisfied the legal department and at the same time allowed the programmer to work with a representative code sample in order to produce the MinImpact product.

3.2.2 Gathering and delivering the code

The author worked with technical leads from several different areas to gather representative code. The code was delivered to the programmer. Some of these same technical leads were used during the pilot phase of the evaluation of the design of the program. There was an iterative approach used from this point on to refine the program.

3.2.3 System Development.

The first version took approximately two weeks elapsed time to write. Attached as figures 3 and 4 are two screen shots. This was the email accompanying the initial release:

Pre-Requisites:

- .Net 2.0 Framework

• Visual SourceSafe Client

• At least one database defined (not one that connects via HTTP) with the client (This is a requirement because the tool searches the registry for defined databases)

Usage:

1st: Add a Project

Upon load the tool searches the registry² you'll notice two data grids one on top of the other, the top is for displaying defined "projects". To define a project click "Add Project". Select a database from the drop down, input the path inside the db to the project you want (e.g. $/SomeProject) along with entering the path you want to save the sources in. You'll notice a checkbox. If that's checked it uses your current logged on credentials to authenticate you with the SourceSafe database, you may also enter alternative credentials (*note* this will be stored in plain text). The user you select needs only to have read permissions for the projects you wish to search and update. Click "OK". The project should now appear in the first data grid.

2nd: Latest Sources

For each project you've defined you'll notice a checkbox in the data grid marked "Include Project". With this field checked it will be included for

² The location is HK_CURRENT_USER\Software\Microsoft\SourceSafe\Databases
searching and updates, make sure the projects you wish to get the latest sources for are checked, and then click the "Get Latest" button. You'll notice shell windows popping up. For each project the first time it will ask you if you want to make this the default directory, if you don't say yes it will continue to ask you this question, press Y and then return and it will continue to retrieve the latest sources. The files it checks out are read only by default, as it uses the "Get" facility and not the "Check out" facility, which should allow us to search the files while not interfering with the development.

3rd: Searching

In the main form you'll see a textbox marked "Search Expression" in this field you may enter any valid regular expression that you wish to use for searching. Once you've entered your query, make sure the projects you wish to search are checked for inclusion and then click "Search". As files are processed you should see the status bar change informing of directory and file changes as well as a progress bar (which may or may not be accurate). Should matches be found you will see them appear in the second data grid. During searching you may cancel at any time by hitting the search button which should read "Cancel" at the time. When finished you'll see notification in the status bar indicating the amount of time spent searching, along with the number of matches found.

4th: Advanced Searching
There are a few more options available to you for advanced searching purposes. You'll see a file type button, by default all files (*.*) are searched with only a few excluded file types, in the file types dialog you'll see groups of check boxes that allow you to specify the types of files you wish to search (mostly an incomplete list) along with a textbox that is used to specifically exclude certain undesirable filetypes that aren't needed for your search. Also you'll see a checkbox for "Contextual" this allows for Multiline regexps (particularly helpful if looking for a call with many parameters that may be separated across newlines). Right now the "context" is only 3 lines but it's easily made configurable and can quickly be added. Finally there is a checkbox for "Ignore case" which allows for case insensitive searching, straightforward.

5th: Viewing Matches

For each match found in the data grid you'll notice the line number, filename, and excerpt. By clicking any of the fields a dialog pops up and allows you to see the full excerpt.

Notes:

The software is written in C# and I opted to write my own "grep" api so I could have easy access to the matches instead of having to parse the output from an external process. The search runs in its own thread with events calling back to update the UI. There's plenty of room for improvement and
lots of basic features that still need to be added (like deleting projects, adding user defined databases, "generic" projects for searching random directories, etc). Not to mention most of the UI pieces need parameter checking to prevent unintended results, along with error checking throughout other pieces of the code. The software of course makes some assumptions, it depends on ss.exe (the cli client interface) and it expects it to be in c:\Program Files\Microsoft Visual SourceSafe\ so if you get an error on check out it could be related to that. Another missing feature is that SourceSafe 2k5 allows for connecting to remote databases via HTTP, I've not setup a database to do that so I wasn't able to add that feature, but connecting to local and remote (smb/cifs) shares works just fine.

At this point, we began the evaluation process with the technical lead(s), one of whom took a major role in this process.

3.2.4 Design and Evaluation

During a five-month period, the developer worked closely with the author and several managers who were aware of the problem we were trying to solve with this tool. Enhancements were made and problems were fixed during this time. Teleconferences ensured that communication was timely and modifications were made to the code as a result of those teleconferences and resultant email explanations. The developer sent several iterations of the code as he worked through bugs and problems that were
uncovered due to the differences in environment between Tim’s development world and the company.

In order to get the developer involved in how these iterations work in a real world environment, he was responsible for recording minutes to the teleconferences and sending them on to the team. Attached as Appendix B are these minutes.

As the pilot progressed, Tim made modifications as requested. Here are examples of some of his iterations, addressing needs from the manager’s wish list:

I've put together another release, you can find it at:
http://www.cs.kent.edu/~tfontain/MinImpact-0.0.2.zip

Three major changes in this version:

- Treeview for project listings -- this allows you to drill down what directories specifically you want to search through. However, if you have the top level directory selected, it will search all sub-directories even if you don't have them checked.

- Treeview for match listings -- each project searched gets its own top-level treenode and then matches with relative paths are shown under that.

- After the end of a search you're prompted to save the results, you need to specify the name and path for where you want the report to save. For now you must specify the csv extension.
I've added prompts if it can't find the SourceSafe executable, and migrated all configuration options to one xml file (as a by product you'll have read any projects). Two exception handlers that will write their messages to files and prompt you with the message and the location of the file. So, if you have problems running this version you should be able at least to give me the full output of the exception from the file.

Next iteration:

Three major changes in this version:

- Treeview for project listings -- this allows you to drill down what directories specifically you want to search through. However, if you have the top level directory selected, it will search all subdirectories even if you don't have them checked.

- Treeview for match listings -- each project searched gets its own top level treenode and then matches with relative paths are shown under that.

- After the end of a search you're prompted to save the results, you need to specify the name and path for where you want the report to save. For now you must specify the csv extension.
3.2.5 Interface

An interface to the SourceSafe system that accomplished the following was developed:

- Copied on demand all source code into a central repository
- Accepted an input string that then produced a list of all references to that string
- Handled errors gracefully
- Referenced the area of code from which the string was found

3.3 Implementation details

Knowing that it would be a non-trivial task to get the technical teams to use the tool, we sent out a survey to 60 people to garner interest in MinImpact. These results indicated that there would be interest and enthusiasm for using the tool. The survey and its results are attached as Appendix A. All 60 people who answered the survey were invited to a meeting to discuss MinImpact. We spoke to managers ahead of the meeting to assure there would be no push-back to taking an hour and a half out of their people’s day.

This email was sent to the managers:

“You all should have been asked to fill out a survey asking you questions about use of a development tool for analyzing the impact of changes to common code. I have had a student from Kent State writing this utility for me. Sreeni has been helping me test it and refine it and I am ready to present it to everyone.”
I am sending you and all of your people invitations for a meeting next Wednesday morning in the amphitheater to discuss this tool. These invitations are also being sent to testers, as I feel they should have first hand knowledge of what the tool can do.

If you do not want your folks to attend (or if you want some folks to attend and not others), please let them know it is okay to decline the invitation.

If anyone would like a short introduction to the tool prior to the meeting, please feel free to give me a call.

You are all also being invited to the meeting.

Thanks!

Alice”

Tim was invited to the company site and the meeting was held. Approximately 50 people attended. Tim presented a demo of the tool during the meeting and folks had suggestions and comments. One of the goals of the meeting was to get developers and their managers to commit to using MinImpact for a short trial period. Out of that population, five developers offered to help with the analysis. Out of those five
developers, four actually made use of the tool. As a result of the feedback from those users, Tim enhanced the tool further approximately 2.5 months later.

All told, there were 5 or 6 iterations of MinImpact, all revised to fix bugs or to add features requested by the developers who were using the tool. The process of writing and revising the program took place over a 7 month period of time.
CHAPTER 4

Case Study

All volunteers installed MinImpact and we worked through a small number of technical problems. Code was checked out, read-only, from SourceSafe and placed on a readily accessible drive for purposes of running MinImpact.

4.1 Feedback on Use of MinImpact

After the five volunteers (Wes, Loganya, Sreeni, Leigh, and Angie) had been involved for several weeks, we elicited comments from the group, using the following questions:

1. How many times did you use the tool last week?
2. Specifically, what did you use the tool for (which query did you enter)?
3. Did you discover anything useful? If so, what? If not, why not?
4. Do you have any suggestions for the tool?

We received answers from four of the volunteers, outlined below. Angie never responded. The summary of the responses are in Table 1. Actual responses are found attached as Exhibit E. Tim created a new version incorporating some of their requests and sent it along for their use.
<table>
<thead>
<tr>
<th>Name</th>
<th>How many times did you use the tool?</th>
<th>What did you use it for?</th>
<th>Did you discover anything useful?</th>
<th>Do you have any suggestions for the tool?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leigh</td>
<td>3</td>
<td>searching for data fields and where they are used</td>
<td>yes – I found what I needed</td>
<td>2 defects; 7 enhancement requests; one compliment</td>
</tr>
<tr>
<td>Sreeni</td>
<td>2</td>
<td>Finding a stored procedure used in two applications</td>
<td>Yes – but I had to search 2 or 3 times</td>
<td>one enhancement request</td>
</tr>
<tr>
<td>Wes</td>
<td>1</td>
<td>Can’t remember – regular expression feature was nice</td>
<td>yes</td>
<td>5 new feature requests</td>
</tr>
<tr>
<td>Loganya</td>
<td>2</td>
<td>Searching for a particular property that I was going to change</td>
<td>Yes – and the property in question was only in a couple of places</td>
<td>General complaint about incompatibility of tool with development environment</td>
</tr>
</tbody>
</table>

### 4.2 Changes in this release:

- Save reports more than at the end of the search
- Pivot report fields
- Add option to [in|ex]clude the text snippet in the report
- Trim whitespace from beginning and end of snippet
- Change report panel to be Project/File/Matches to see matches per file to reduce clutter
- The suggestion that is missing:
  - Saving a report on a subset of the results
It can be seen that users’ suggestions were considered and subsequent action was taken. Based on the number of suggestions and our ability to act on them, we met our condition of producing a tool for which the users felt ownership. Without this partnership in place, we feel users would have been very resistant to use of the tool.

4.3 Case Study Findings

In general, the users felt there was value in the tool. It is difficult to get users in a real world environment to continue to use a tool when its intended purpose (impact analysis) is not a rewarded behavior in the organization.
CHAPTER 5

Summary and Conclusions

The goal of the case study was to produce the MinImpact tool, conduct a pilot, make improvements to the tool, and then make it available to a subset of developers at the company. A second goal was that the developers use the tool and determine that it was a useful way to perform a lightweight impact analysis for small changes (especially bug fixes) that they were conducting as part of their jobs.

Both of the goals were met. Some of the learnings of the case study included:

• A lightweight impact analysis must be so easy to use that it adds virtually no time to the development process.

• The tool must be so transparent to the current development environment that it doesn’t interfere with day-to-day development tasks.

• One side effect of this project was that the author discovered that not all code was under configuration management. Specifically, stored procedures are not in configuration management, are not versioned, and authors are not designated. Modifications to stored procedures caused some of the most insidious bugs in production due to their shared use across multiple systems, yet it was difficult if not impossible to search for them across the source code.
Conducting a dedicated pilot that is enforced across the life of a specific maintenance project may yield more consistent results than a case study and should be considered. One programmer pointed out to us that “this feature is already available to some degree in SourceSafe” and yet no one was using it. This suggests that the email communication route is preferred to a more formal method and it may be more beneficial to get a “forced” pilot into place.

Following up on the information provided by the tool should be part of the pilot. This may very well add time to the project, depending upon the impact analysis the tool exposes. After that period, management can analyze the data provided and determine which errors have been avoided due to its use. A side effect of the use of the tool will be to provide management with a measure of the complexity of the systems under development. This understanding will help in estimations provided for future project work and may well aid in the design of later systems.

The MinImpact output must be a required deliverable during the pilot from the developer performing the coding. These output reports can be investigated by management so that determination of whether the change is really as “small” as initially estimated is accurate. In addition, the report’s pages can be assigned to individual team members to change the code in question. Determination of what must be fixed, recompiled, retested, and re-released can be made via management of the report pages. If the change needs to go to production immediately, at least the reports exist for later investigation into the possible repercussions of the emergency change and may be assigned as time permits in the schedule.
APPENDIX A

Survey

This survey was given to 78 developers, testers, and managers to generate interest in MinImpact and to assure user requirements were taken into consideration in the design of the tool.

Survey Results for Impact Analysis

Launch Date: 1/25/2007 5:45:28 AM  
Total Participants: 78  
Total Respondents: 60

1. What best characterizes your position at the company?

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of Responses</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domestic Contractor</td>
<td>17</td>
<td>28%</td>
</tr>
<tr>
<td>Employee</td>
<td>32</td>
<td>53%</td>
</tr>
<tr>
<td>Onshore partner Employee</td>
<td>4</td>
<td>7%</td>
</tr>
<tr>
<td>Offshore partner Employee</td>
<td>7</td>
<td>12%</td>
</tr>
<tr>
<td>Total no of Respondents</td>
<td>60</td>
<td>100%</td>
</tr>
</tbody>
</table>

2. What best characterizes your role at the company?

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of Responses</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>28</td>
<td>47%</td>
</tr>
<tr>
<td>Tester</td>
<td>13</td>
<td>22%</td>
</tr>
<tr>
<td>Manager</td>
<td>12</td>
<td>20%</td>
</tr>
<tr>
<td>Others, Specify</td>
<td>7</td>
<td>12%</td>
</tr>
<tr>
<td>Total no of Respondents</td>
<td>60</td>
<td>100%</td>
</tr>
</tbody>
</table>

Responses for Others, Specify

1. DB Admin / Tester / Developer
2. Project Leader (Tech Lead)
3. Tech Lead (Microsoft)
4. CE application support.
5. Technical Lead
6. Technology Support
7. support

3. **When you develop (test) code, how do you know if the change impacts other systems/modules/applications?**

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of Responses</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Don't know</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Depend upon Technical Lead to tell me</td>
<td>19</td>
<td>32%</td>
</tr>
<tr>
<td>Educated guess</td>
<td>12</td>
<td>20%</td>
</tr>
<tr>
<td>Others, Specify</td>
<td>29</td>
<td>48%</td>
</tr>
<tr>
<td><strong>Total no of Respondents</strong></td>
<td>60</td>
<td>100%</td>
</tr>
</tbody>
</table>

**Responses for Others, Specify**

1. Tech Lead, educated guess. database analysis
2. Unit / Regression Testing
3. documentation and design docs
4. Due Diligence
5. experience, and checking with application owner
6. Depends on the project
7. During the Requirement analysis/Design phase will analyse the Impacts on other systems also
8. Requirement document and other documents, used to consult with application owners
9. Through the business requirement document
10. Analyzing the modules
11. By having clear knowledge of whole system, we test the developed code effect on other modules.
12. We test the developed code effect on the other modules. Prior to start coding we analyse the impact of this changes.
13. Based on discussion with Other Leads
14. By searching other modules of the applications for usage of the units/code pieces being modified in current project
15. Ask around
16. Research/ask other developers
17. Business Requirements and the lead
18. Regression and unit testing
19. From working with the system, I’ve learned
20. Research code and ask peers.
21. I talked to the application owners and respective Tech Leads
22. ask others
23. I depend on my expertise
24. Various reasons. Experience, other associates, good planning.
25. Look through code base to see what other applications utilize the code being modified
26. NA
27. search code base for other impacted modules
28. own knowledge
29. Call app owners

4. **Have you ever changed (or tested) code for one system/application/module and experienced an unanticipated impact on other systems/applications/modules?**

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of Responses</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>36</td>
<td>60%</td>
</tr>
<tr>
<td>No</td>
<td>24</td>
<td>40%</td>
</tr>
<tr>
<td><strong>Total no of Respondents</strong></td>
<td><strong>60</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>

5. **If you determine that your change impacts other systems/modules, do you test all impacted modules?**

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of Responses</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>51</td>
<td>85%</td>
</tr>
<tr>
<td>No</td>
<td>9</td>
<td>15%</td>
</tr>
<tr>
<td><strong>Total no of Respondents</strong></td>
<td><strong>60</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>

6. **Would you be interested in evaluating a tool that helps you determine the impact of your changes on other systems/applications/modules?**

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of Responses</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>52</td>
<td>87%</td>
</tr>
<tr>
<td>No</td>
<td>8</td>
<td>13%</td>
</tr>
<tr>
<td><strong>Total no of Respondents</strong></td>
<td><strong>60</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>
APPENDIX B

Email meeting minutes

Tim, the student programmer, was responsible for assuring that he was creating and delivering the correct features in MinImpact. Therefore, he was responsible for recording user concerns in the beginning phases of the project. Below is an example of meeting minutes he recorded during one of the initial meetings.

The conference call overall was very helpful, it allowed me to asses the weaknesses in my GUI design and implementation. Very generally speaking I need to provide more user feedback to explain the processes going on and to be able to clearly define the nomenclature I use for particular fields and buttons. I was happy once we successfully had the application working as intended and that it did appear to be useful.

Below I've summarized the highpoints that need attention so the application can be used efficiently and more widespread, some of them I had mentioned in the README, others are deficiencies on my part or features that are needed. User feedback and error checking are common themes that I admit I need to work on.

Error Checking:

* Add proactive parameter checking for form entries
  - Make sure to provide feedback to user to inform them of problems and possible solutions

* Handle permission errors in Grep code
  - Again, provide user with feedback

* General exception handling throughout the application
Filetypes:

* Provide more groups and proper associations of file types
  (sql, delphi, [word, excel (maybe)], csv, etcetera)
- Allow user to specify more accepted extensions
- Serialize added file types for persistence across application usage
* Error checking and user feedback

[Sub]Projects:

* Treeview style traversal (think folder browser dialog with checkboxes)
  - Add Project Dialog should provide means of project selection by discovering available [sub]projects
    -- Entering projects or subproject paths should check for errors and provide feedback
    -- By default added projects should not be included for searching or updates
    -- Possibly add option for generic "projects" that are user defined paths and not actual projects under source control
  - Project Dialog should utilize treeview to allow further drill downs of [in|ex]cluded paths/directories for searching
    -- Also add context menu for allowing Delete/Modify/Update of specific [sub]projects

Searching:

* Perhaps add an option that parses .scc files and allows for searching of files only under source control

Match Dialog:

* Selecting a found match should load entire source file (iff the file is not actively being searched) and highlight the excerpt of said match
- Later glitz can be added with possible syntax highlighting

Report Export:

* Provide a means for easy export of matches for a particular query to useful file format so matches can be analyzed ex post facto

General User Interface:

* Reposition Buttons and Checkboxes to allow clearer association with actions
* Fixing tab stops so application can be used without mouse
* More accurate button labels that better define actions
* Spinner for increasing the size of Contextual searches
* More and More and More user hints and feedback to guide user through application
* Again and Again remember to add feedback for error checking!

Any particular visual dialog changes as far as reordering of the buttons and checkboxes in future releases it would be helpful if mockups could be made to show what you'd like to see. If I've missed anything else we spoke about please email me the reminder.

Thanks,

Tim Fontaine
APPENDIX C

Email with initial requirements

This email represents the initial requirements shared with Tim, the student programmer.

It was great meeting you on Tuesday, Tim. I look forward to working with you. Below please find my synopsis of what we discussed:

1. You will create a tool that will search all projects within SourceSafe for a given string/function call/module call. The string submitted may require the insertion of a regular expression by the user to minimize false positives.

2. In order to do this, you need a small, representative sample of code from the projects within our SourceSafe structure. Small can be one or two dozen modules/programs.

3. The tool will work on various programming languages, including, but not limited to: SQL scripts/stored procedures, Delphi, Visual Basic, C++, etc. Documentation (e.g., Microsoft docs) are not to be included in the scope of the search.

4. The SourceSafe repository sits on a single server or is accessible from a single server.

5. You will probably need to check out a read-only copy of the latest of all code and place in a "sandbox" area to run your utility. Hence, you will also need to provide a utility that does that checkout on a regular basis (quickly), with the user specifying the location in which to place everything.

6. I will check with legal to get the correct confidentiality agreement for you to sign and email it to you.

7. I will get a better idea of the number of lines of code contained in SourceSafe and send to you.
I just remembered - there is some funky Edify code contained within SourceSafe that we probably don't have to worry about. I will find out the location so that it can be excluded, or alternatively, you can provide me an interface that allows for a "search of everything except this 1 project, or 2 projects, or 3 projects or 1 directory within this project ..."

I think that's it ... Let me know if you remember anything else.

Thanks,

Alice
APPENDIX D

Use Case specifications

In order to assure that the student programmer received a more formal set of specifications, these use cases were written after the initial conversation and follow-up email.

MinImpact System

Use-Case Specification

*Search for a string within a context of one or more Projects’ worth of source code*

Version 1.0

Revision History

9/15/2005 Draft Alice Lewis

1. Brief Description
   This use case allows the builder to search for a string within the source code that is contained in all projects found in SourceSafe. The actor is either the builder or the developer.

2. Flow of Events
   The use case begins when the builder selects the “search for string” item from the menu.

   Basic Flow – Choose project(s)
   The builder selects “search for string”
   The system displays a drop down box containing all projects currently in SourceSafe and an “all projects” choice
   The builder chooses one or more projects to search from the drop-down, using <ctrl> mouse click as appropriate to choose multiple projects.
   If no projects are chosen, no further action is taken.
Basic Flow – Enter Search String
Pre-condition – a project or list of projects have been chosen.
The builder can enter a Search String
A text box of length 100 appears
The builder enters a string of 100 or fewer characters
Any character that is acceptable in the following languages will be acceptable for the string: vb, vb.net, asp, html, Delphi, SQL stored procedures, <I will provide the complete list>.

Basic Flow – Output after string is entered
The builder is presented with a list of all files which contain the string in question, with the most recently checked-in file appearing on the list. The list will present <project name><file name><line in file containing string>. The output will appear on the screen, but an option will allow the builder to send output to a text file.
The builder can click on the <file name> and pull up the file itself, with the lines in which the string appears highlighted.
If the list of files is longer than a page, the user will be able to scroll down the page to see all files.

3 Special Requirements
The system must span all SourceSafe projects, including potentially those that contain documentation only (Word documents).
The system must be accessible to everyone who has access to the SourceSafe system in a view only mode.
Developers from remote locations who can access SourceSafe must also be able to access MinImpact.

4 Preconditions
The builder/developer should have read access to all projects in SourceSafe – or at least the ones that appear on the drop-down list.
The system must run on a Windows xp based workstation, preferably IE browser based.
The system should not allow write access to any of the files in question.
The system should allow for multiple users to access the system at one time with no adverse effects on any user.
If the file that MinImpact has found has been checked out of SourceSafe by a user, the system will access the latest checked in version for reference and note that it is currently checked out and by whom.

5 Post conditions
The output will appear to the screen as well as a user-designated text file.
APPENDIX E

User Feedback on Use of MinImpact

A mini-survey was conducted several weeks into the case study. These questions were asked. In addition, unsolicited feedback was also gathered and is included in this appendix.

5. How many times did you use the tool last week?
6. Specifically, what did you use the tool for (which query did you enter)?
7. Did you discover anything useful? If so, what? If not, why not?
8. Do you have any suggestions for the tool?

Answers are found below.

Leigh:

1. I've used the tool 3 times
2. I used it for searching for data fields and where they are used
3. Found what I needed
4. I noticed a couple things - if the first project in the project list isn't checked you get a pop-up error saying you need to select a project. But when you click ok - it works anyways. I think the ignore case should default to checked - most searches don't care about the case and if you forget to check it - it's annoying. Just so you don't think I'm all negative, I really like that when you click on one of the lines it opens up the unit so you can see the actual code.
Sreeni:
1. I’ve used the tool two times
2. Regarding finding a stored procedure used in Web, PSL
3. Yes, But we have to search two three times
4. If it is possible to get dependencies as well

Wes:

I think it has some value. I forget what I was searching for now, but the regular expression feature was nice.

I will use it more in the next few weeks as I start the next portion of SmartSteps Phase III.

Loganya:
1. I think I used the tool 2 times
2. can't remember exactly, but one of the times was to search for a particular property that I was going to change
3. as I recall it was useful - I found that the property I looked for was only in a couple of places
4. no suggestions this week”

Unsolicited feedback:
1. I don't have ASP.NET 2.0 in my machine. If I install the same I can not run <our company website>.com. So, I have to install the same in different machine having ASP.NET 2.0
2. I'd like there to be a save button on the form rather than the pop-up at
the end. I saved my results but I don't remember where they went and now I either have to run it again or search my whole machine for the file.

3. I'd also like the ability to highlight the results area and copy them that way.

4. I found something else 'I wish'. When I save the report, it doesn't save the text of the line where the hit was found. I wanted to be able to save the results exactly as they are displayed and be able to search thru them. All the report shows me is the unit and line number. The screen shows the actual line. This is the screen output - I wanted the 'select_prt_sql^.use_elections := false' to be in the saved report too.

5. I have taken the MinImpact app for a test drive. It looks good so far, but I have some thoughts about usability. (For reference, I am using v0.0.5). The output gets somewhat cluttered when the tool finds multiple instances of the search criteria in one file. Is it possible to collapse the information per file as follows:

- In the results display, eliminate white space at the start of the line (spaces, tabs, etc.)
- In the "Current Match" dialog box, close it by typing <ESC>.
- Pivot the information at the top of the report file (.csv) as follows <example proprietary and not included>:
- Add a button to export the current results to a report file. (Currently,
the only way to produce the report seems to be to answer "Yes" in the message box that comes up when the tool completes a search.)
APPENDIX F

Input to MinImpact

Input to MinImpact®

MinImpact®

Search:  Code    Data

Population:

Projects: [drop down box and ability to "select all"

Dates: [date ranges for each project or ability to "select latest"]
APPENDIX G

Output from MinImpact

MinImpact® Search for "part_addr" revealed:
Page 1 of 32
Application: PSL
Component: AddressMod
15 [contents of line 15]
...
25 ... part_addr ...
...
35 [contents of line 35]

AddressMod last modified by CC on 2/15/2004
APPENDIX H

MinImpact, Initial Screen
APPENDIX I

MinImpact, Choose Projects from SourceSafe
APPENDIX J

MinImpact, Choose File Types
APPENDIX K

MinImpact, search for “ray”
APPENDIX L

MinImpact, found “ray”
APPENDIX M

MinImpact, all results of searching for “ray”
APPENDIX N

MinImpact, saving result of searching for “ray”
APPENDIX O

Saved results of searching for “ray”

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>3/25/2009</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>DIYProjectMiniProject</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expression</td>
<td>Vector</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construction</td>
<td>FALSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LinearVector</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hits</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
void setVector(Vertex v) {
    // Vector initialization
}

Vertex getVertex() {
    // Return vector
}

Ray getRay() {
    // Return ray
}

IntersectionPoint setIntersectionPoint(Ray ray) {
    // Set intersection point
}

IntersectionPoint getIntersectionPoint() {
    // Get intersection point
}

Ray getRay2() {
    // Return ray2
}
```

```
// Example code
Ray ray = new Ray();
IntersectionPoint intersectionPoint = ray.getIntersectionPoint();
```
REFERENCES


