Experiences with tracking the effects of changing requirements on Morphbank: A web-based bioinformatics application

Subhajit DATTA
Singapore Management University, subhajitd@smu.edu.sg

Robert van Engelen
Florida State University

David GAITROS
Florida State University

Neelima JAMMIGUMPULA
Florida State University

Experiences with Tracking the Effects of Changing Requirements on Morphbank: A Web-based Bioinformatics Application

Subhajit Datta [*]
Robert van Engelen
David Gaitros
Neelima Jammigumpula
School of Computational Science
Florida State University
Tallahassee, FL 32306, USA
{subhajit, engelen, gaitros, jammigum}@scs.fsu.edu

ABSTRACT

In this paper, we present a case study of applying the metrics *Mutation Index*, *Component Set*, *Dependency Index* [1] on *Morphbank*– a web based bioinformatics application – to track the effects of changing requirements on a software system and suggest design modifications to mitigate such impact. Morphbank is "an open web repository of biological images documenting specimen-based research in comparative anatomy, morphological phylogenetics, taxonomy and related fields focused on increasing our knowledge about biodiversity" [2]. This paper discusses the context of the case study, analyzes the results, highlights observations and learning, and mentions directions of future work.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, process metrics*

General Terms

Algorithms, Management, Measurement, Design

Keywords

Analysis, Design, Metrics

1. INTRODUCTION

[*]Email correspondence may be directed to this author at subhajit@scs.fsu.edu.

Managing the effects of changing requirements remains one of the greatest challenges of enterprise software development. Based on Fowlers' insights [3] we note some of the characteristics of enterprise software systems to be: persistent data, concurrent access of data, lot of user interface screens, need to bridge "conceptual dissonance" between the demands of diverse stakeholders, etc.

[1] derives a set of metrics and suggests a methodology to measure the effects of requirement changes from one iteration to another, and to facilitate design decisions to mitigate these effects. To evaluate this approach, we undertook a case study of the Morphbank project. The project was suited to the technique as it underwent (and is still undergoing) several cycles of user driven requirement modifications. (We describe the context of Morphbank in a later section.)

The case study was an interesting exercise, giving us new insights on the applicability of the metrics and scope of further work. In the next section we present an overview of the metrics, followed by a brief description of the project, scope of our case study, results and interpretations, directions of future work, and conclusion.

2. OVERVIEW OF THE METRICS

As stated earlier detailed derivation of the metrics and discussion on the methodology are covered in [1]. We give an overview below.

The metrics are developed on the idea that a software system fulfills user requirements through the interaction of its components. Intuitively, the *Mutation Index* measures the extent to which a requirement changes from one iteration to another; *Component Set* is the collection of components needed to fulfill a particular requirement and is used in the calculation of the next metric; *Dependency Index* measures the level of interaction between a particular requirement's components with other components in the system.

The following assumptions were made:

- The context of the discussion is *functional* requirements.
- By reviewing a particular requirement, an experienced analyst is able to recognize whether it concerns the

$Display$, $Processing$ or $Storage$ aspects of the system. $Display$ subsumes all features of the user interface and interaction facilities between the user and the system. $Processing$ is any nontrivial modification of information performed by the system. $Storage$ includes all activities associated with persisting information and accessing such information.

- When a requirement changes, the change can affect $Display(D)$, $Processing(P)$ or $Storage(S)$; singly or collectively. Thus, between iterations, each changing requirement, R_n is attributed a $Mutation\ Value$ $MV(n)$ of D, P or S; or any of their combination.

- The $Display$, $Processing$ and $Storage$ aspects may be associated with the three basic types of components; *Boundary*, *Control*, and *Entity* [4].

- The metrics address requirement changes between iterations; identification of current, previous and next iterations is implicit in the discussion.

- *System* refers to the software under development along with its interfaces. *Component* refers to logical/physical entities whose interaction is necessary for the working of the system.

For a system, let every requirement identified during requirement elicitation be uniquely named as $R_1, R_2, R_3, ..., R_n$. Between iterations I_{n-1} and I_n each requirement is annotated with its $Mutation\ Value$; a combination of the symbols D, P and S. The symbols stand for:
$D \equiv Display(1)$
$P \equiv Processing(2)$
$S \equiv Storage(3)$

The parenthesized numbers denote the $Weights$ attached to each symbol. The combination of more than one symbol signifies the addition of their respective $Weights$, thus:
$DP \equiv 1 + 2 = 3$
$DS \equiv 1 + 3 = 4$
$PS \equiv 2 + 3 = 5$
$DPS \equiv 1 + 3 + 2 = 6$

Evidently, the order of the combination is irrelevant here, i.e. $DP \equiv PD, DPS \equiv SPD$ etc.

It should be pointed out the $Weights$ as assigned above are different from [1]. Since Morphbank is a database intensive application, we thought it fit to assign the highest $Weight$ to the $Storage$ aspect vis-a-vis the other ones. Similar changes may be introduced when other applications are studied, depending on their scope and complexity.

The $Weight$ assigned to each category of components – $Display$, $Processing$ and $Storage$ – is a relative measure of their complexities. $Storage$ components encapsulate the access and updating of application data stores – a key functionality of Morphbank – $Processing$ components embody application logic – not very involved in this project – and $Display$ components deal with user interface, which in this case is of the level of complexity usual for a web based project. Accordingly, $Display$, $Processing$ and $Storage$ have been assigned the $Weights$ 1, 2 and 3 respectively. Exact values of $Weights$ may be varied from one project to another; the essential idea is to introduce a quantitative differentiation between the types of components.

2.1 Mutation Index

The Mutation Index $MI(n)$ for a requirement R_n is a relative measure of the extent to which the requirement has changed from one iteration to another in terms of the components needed to fulfill it.

Expressed as a ratio, the $MI(n)$ for requirement R_n :

$$MI(n) = \frac{The\ Mutation\ Value\ for\ R_n}{The\ maximum\ Mutation\ Value} \quad (1)$$

Thus, if at iteration I_n, Requirement R_m has been assigned a $Mutation\ Value\ MV(m) = DS$ with reference to iteration I_{n-1}, $MI(m)$ is calculated as :

$$MI(m) = \frac{DS}{DPS}$$

$$MI(m) = \frac{4}{6}$$

$$MI(m) = 0.67$$

Intuitively, if change in R_m can only affect the $Display$ aspects of the system, the corresponding $MI(m) = D/DPS$ $= 1/6 = 0.17$, which is less significant than the changes affecting only $Processing$, i.e. $MI(m) = P/DPS = 2/6 =$ 0.33 or only $Storage$, i.e. $MI(m) = S/DPS = 3/6 = 0.5$.

At the boundary conditions, if a requirement has not changed from one iteration to another, the $Mutation\ Value$ is 0 and $MI(m) = 0/6 = 0$. And, if all of $Display$, $Processing$ and $Storage$ aspects will be affected by changes in the requirement, the $MI(m) = 6/6 = 1$. $MI(m)$ for a requirement R_m can vary between these extreme values.

2.2 Component Set

The Component Set $CS(n)$ for a requirement R_n is the set of components required to fulfill the requirement.

As a system's design evolves, components fulfilling a particular requirement typically undergo several cycles of refinement. So for each iteration, a requirement may have *Component Set* different from preceding or succeeding iterations. Let the following components combine to fulfill requirement R_n :
$C_B \equiv Set\ of\ Boundary\ components$
$C_C \equiv Set\ of\ Control\ components$
$C_E \equiv Set\ of\ Entity\ components$
$C_X \equiv Set\ of\ helper,\ utility\ and\ other\ components.$

Then, the $Component\ Set\ CS(n)$ for R_n is defined as,

$$CS(n) = C_B \cup C_C \cup C_E \cup C_X \quad (2)$$

2.3 Dependency Index

The Dependency Index $DI(m)$ for a requirement R_m is a relative measure of the level of dependency between the components fulfilling R_m and those fulfilling other requirements of the same system.

For a set of requirements R_1,R_2,...,R_m,...,R_{n-1},R_n, let
$Y = CS(1) \cup CS(2) \cup ... \cup CS(n-1) \cup CS(n)$

For a requirement R_m, $(1 \leq m \leq n)$, let
$Z(m) = (CS(1) \cap CS(m)) \cup ... \cup ((CS(m-1) \cap CS(m)) \cup$
$((CS(m) \cap (CS(m+1)) \cup ... \cup ((CS(m) \cap (CS(n))$

$Z(m)$ denotes the set of components that play a part in the implementation of R_m, as well as some other requirement, say R_j.

The $DI(m)$ for requirement R_m is expressed as the ratio:

$$DI(m) = \frac{|Z(m)|}{|Y|} \qquad (3)$$

(For a set S, $|S|$ is taken to denote the number of elements of S.)

The approach in [1] examines the metrics values in the light of one another as development proceeds through iterations and incremental releases. At iteration k, the *Mutation Index* indicate how much the requirements have changed from iteration $(k-1)$. The *Dependency Index* values for iteration $(k-1)$ convey how much the requirements are dependent on one another for their fulfillment. Analyzing the $MI(n)$ and $DI(n)$ values for some R_n for these two iterations, the level of design modification needed to implement the changed requirements can be gauged. For frequently changing requirements, design choices that will bring down their respective $DI(n)$ values will be most amenable towards absorbing future changes.

We will now discuss the application of the metrics on Morphbank.

3. BACKGROUND OF MORPHBANK

Morphbank serves the biological research community as an open web repository of images. "It is currently being used to document specimens in natural history collections, to voucher DNA sequence data, and to share research results in disciplines such as taxonomy, morphometrics, comparative anatomy, and phylogenetics" [2]. The Morphbank system uses open standards and free software to store images and associated data and is accessible to any biologist interested in storing and sharing digital information of organisms. Morphbank was founded in 1998 by a Swedish-Spanish-American consortium of systematic entomologists and is currently being developed and maintained by an interdisciplinary team at a United States university.

Morphbank's principal goal lies in developing an web-based system to support the biological sciences in disciplines such as taxonomy, systematics, evolutionary biology, plant science and animal science. Morphbank facilitates collaboration amongst biological scientists by allowing for the sharing of specimen images, annotating existing images, remotely curate natural history collections, and build phylogenetic character matrices.

Morphbank provides features such as browsing, searching, submitting, editing, annotating of biological specimen data. Since the Morphbank was taken up by the current development team, the project has passed through releases 2.0, 2.5, with releases 2.7 and 3.0 being planned.

The key element of Morphbank is supporting a collaborative environment. Thus expectedly, the requirements undergo frequent changes as different groups of users communicate their changing needs. We focus our attention on the changing requirements for the Browse functionality.

4. BROWSE FUNCTIONALITY

Morphbank functional areas can be broadly classified into *Browse, Search, Submit, Edit, Annotate* etc. [5]. Out of these we choose the Browse functionality for our case study. This choice is inspired by the fact that Browse has undergone several requirement changes between Morphbank 2.0 to Morphbank 2.5 and changes are also expected in the future versions. Browse remains by far the most visible of the functional areas; thus user needs undergo frequent modifications. The major requirements under the Browse functional area are listed in Table 1 and their changes noted from Morphbank 2.0 to Morphbank 2.5. We will apply the metrics on these requirements and their changes. The changes can be summarized as: between Morphbank 2.0 and Morphbank 2.5 three new *ways* of Browsing, by Collection, by Image, and by Taxon were introduced, as well as Search facilities were provided from within the Browse interface. The Search feature of Morphbank allows users to find a specific record or a set of records based on a specific input criteria.

As an example, Browse by View screen image is given in Figure 1.

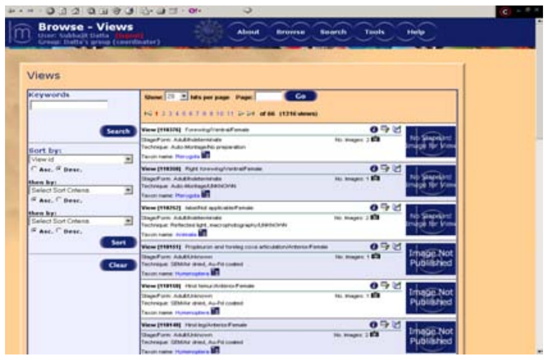

Figure 1: Morphbank Browse by View

5. CODE COMPONENTS

Morphbank uses PHP components and the Morphbank and ITIS (Integrated Taxonomic Information System) [6] databases to deliver its functionality. Table 2 lists the components for each of the Browse requirements for Morphbank 2.0 and Morphbank 2.5. In addition, the following *common* components were used across the requirements:

5.1 Common components for Morphbank 2.0

- config.inc.php,footer.inc.php,head.inc.php, http_build_query.php,mail.php,menu.inc.php,nusoap.php, objOptions.inc.php,pop.inc.php,queryLogFunctions.php, qlODBC.inc.php,thumbs.inc.php,treeview.inc.php, tsnFunctions.php,webServices.inc.php, layersmenunoscript.inc.php, layersmenuprocess.inc.php,template.inc.php
- layersmenu.inc.php,layersmenu.inc.php.orig, layersmenunoscript.inc.php,layersmenuprocess.inc.php, template.inc.php
- annotateMenu.php,datescript.js,layersmenu.js, layersmenubrowser_detection.js,layersmenufooter.ijs, layersmenuheader.ijs,layersmenulibrary.js,layerstreemenu.ijs, layerstreemenucookies.js

Table 1: Morphbank Browse Requirements

Req.ID	Morphbank 2.0	Morphbank 2.5
R_1	Browse by Location	Added search facilities
R_2	Browse by Name	Added search facilities
R_3	Browse by Specimen	Added search facilities
R_4	Browse by View	Added search facilities
R_5	Did not exist	Browse by Collection with search facilities
R_6	Did not exist	Browse by Image with search facilities
R_7	Did not exist	Browse by Taxon with search facilities

5.2 Common components for Morphbank 2.5

- config.inc.php,footer.inc.php,head.inc.php, http_build_query.php,mail.php,menu.inc.php,nusoap.php, objOptions.inc.php,pop.inc.php,queryLogFunctions.php, sqlODBC.inc.php,thumbs.inc.php,treeview.inc.php, tsnFunctions.php,webServices.inc.php, collectionFunctions.inc.php,copyCollection.php, editExtLinks.php,editjavascript.php,editjavascripts.php, ExtLinks.php,imageFunctions.php,postItFunctions.inc.php, showFunctions.inc.php,XML.inc.php,navigation.php

- layersmenu.inc.php,layersmenu.inc.php.orig, layersmenunoscript.inc.php,layersmenuprocess.inc.php, template.inc.php

- annotateMenu.php,datescript.js,layersmenu.js, layersmenubrowser_detection.js,layersmenufooter.ijs, layersmenuheader.ijs,layersmenulibrary.js, layerstreemenu.ijs,layerstreemenucookies.js, date.js,determinationJS.inc.php,extLinks.js,general.js, gotoRecord.js,localityEdit.js,popupdate.js, specimenEdit.js,viewEdit.js

- collectionFilter.class.php,filter.class.php, filters.class.php,keywordFilter.class.php, localityFilter.class.php,resultControls.class.php, sort.class.php,specimenFilter.class.php, tsnFilter.class.php,viewFilter.class.php

There were minor database related changes between Morphbank 2.0 to Morphbank 2.5 but these did not directly affect the Browse functionality. The introduction of the search mechanism within Browse was handled by the code components.

6. CALCULATING THE METRICS

Morphbank 2.0 and Morphbank 2.5 represent incremental releases in the system's evolution. We take iteration 1 (I_1) and iteration 2 (I_2) to be the collection of activities which lead to these two releases respectively.

Based on the discussion in the earlier sections we calculate the *Mutation Value*, *Mutation Index* and *Dependency Index* for I_1 and I_2 in Table 3 and Table 4. (As explained earlier the *Component Set* is used in an intermediate step in the calculation of the *Dependency Index*.) We note the changes to the existing requirements of Morphbank 2.0 to Morphbank 2.5 relate to the *Display*, and *Storage* aspects; as search functionality was added to all of the Browse categories. There was no change in *Processing* as such, only modifications to database access logic and presentation. It may be also underlined each PHP component combines all of *Display*, *Processing*, and *Storage*. Thus change in any one of these aspects necessitates modification of the component.

Table 3: Metrics for I_1 of Morphbank Browse functionality

Req	$MI(n)$	Y	$\|Z(n)\|$	$DI(n)$
R_1	0	40	32	0.8
R_2	0	40	32	0.8
R_3	0	40	32	0.8
R_4	0	40	32	0.8
R_5	-	-	-	-
R_5	-	-	-	-
R_5	-	-	-	-

Table 4: Metrics for I_2 of Morphbank Browse functionality

Req	$MI(n)$	Y	$\|Z(n)\|$	$DI(n)$
R_1	0.67	82	61	0.74
R_2	0.67	82	61	0.74
R_3	0.67	82	61	0.74
R_4	0.67	82	61	0.74
R_5	0	82	61	0.74
R_6	0	82	61	0.74
R_7	0	82	61	0.74

7. INTERPRETATION

The $MI(n)$ values for all the requirements in I_1 is 0, which is expected, as in the very first iteration, there is no previous iteration to measure a requirement change against. We find the $DI(n)$ values for the requirements R_1,...,R_4 are all 0.8. This is due the fact that as per the design, different Browse requirements are implemented by *independent* groups of components. The only shared components across requirements are the so called *common* components listed in an earlier section. Figure 2 shows the variation of the metrics between Morphbank 2.0 and Morphbank 2.5.

So the dependencies across the components are evenly distributed, although the level of dependency is significantly high with uniform $DI(n)$ values of 0.8. The changes in Browse requirements between Morphbank 2.0 and Morphbank 2.5 manifested as additional functionality for R_1,...,R_4 and introduction of the new requirements R_5, R_6, and R_7. As stated above, for all the requirements, the changes were in the *Display* and *Storage* aspects, resulting in the same $MV(n)$ values of $DS \equiv 1 + 3 = 4$, and hence, $MI(n) = 4/6 = 0.67$. Given these $MI(n)$ values for I_2 and the high $DI(n)$ values for I_1, corresponding to each requirement, *it*

Table 2: Morphbank Browse Code Components

Req.ID	Morphbank 2.0	Morphbank 2.5
R_1	index.php,mainBrowseByLocation.php	index.php,mainBrowseByLocation.php, resultControls.class.php
R_2	index.php,mainBrowseByName.php	index.php,mainBrowseByName.php
R_3	index.php,mainBrowseSpecimen.php	index.php,mainBrowseSpecimen.php,resultControls.class.php
R_4	index.php,mainBrowseByView.php	index.php,mainBrowseByView.php,resultControls.class.php
R_5	Not Applicable	index.php,mainBrowseByCollection.php,resultControls.class.php
R_6	Not Applicable	index.php,copyToCollection.php,copyToNewCollection.php, listImageThumbs.inc.php,mainBrowseByImage.php, resultControls.class.php
R_7	Not Applicable	index.php,mainBrowseByTaxonTree.php

is expected a significant amount of change in implementation will be needed to accommodate the modified functionality. Let us examine the extent of code change between Morphbank 2.0 to Morphbank 2.5.

8. VALIDATION OF METRICS BASED INSIGHT

Between I_1 and I_2, the number of Morphbank Browse components increased by more than 102% (40 components in Morphbank 2.0 vis-a-vis 81 components in Morphbank 2.5). Additionally, 45% of the components from I_1 were modified in I_2 (18 of the 40 components of Morphbank 2.0 were changed and deployed in Morphbank 2.5). Figure 3 shows the new, changed and unchanged components between Morphbank 2.0 and Morphbank 2.5. To detect modification of a component, a textual comparison of the corresponding file for I_1 and I_2 was done by the *Examdiff* visual file comparison tool [8]. The number of differences between two versions of a component ranged from minimum of 1 to maximum of 41. This empirical data validates the metric based insight of high *Dependency Index* values indicating need for significant rework even for changes in requirements related to *Display* and *Storage* (as given by the *Mutation Index* values).

We observe the $DI(n)$ values for I_2 are somewhat lower at 0.74 compared to 0.8 for I_1. So the extent of rework for similar mutation of requirements in a subsequent iteration is expected to be lower than that necessitated in I_2. However, as we discuss in the following section, the trend of requirement changes for the Browse functional area may be better served in the long run by a different direction of the design.

9. OBSERVATIONS AND LEARNING

- The metrics in [1] assume a clear *separation of concerns* in the system design: separate components implement *Display, Processing,* and *Storage* aspects of a functionality. But the PHP components in Morphbank 2.0 and Morphbank 2.5 *combine* the implementation of all of these aspects. Although the original approach is based on the standard n-tier architecture of enterprise software systems, the metrics are equally applicable in the Morphbank scenario. This was established in the preceding sections by the close correlation of the prediction from the metrics and the empirical data on the extent of code change between Morphbank 2.0 and Morphbank 2.5.

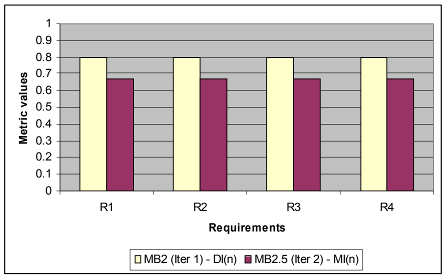

Figure 2: Variation of DI(n) & MI(n) values for Browse requirements between Morphbank 2.0 and 2.5

- The existing Morphbank architecture makes the implementation of each requirement as concentrated as possible amongst a small number of components. This uniformity is reflected by the same $DI(n)$ values for all the requirements in both in first and the second iteration. A high degree of component independence is expected to insulate the components to a large extent from the effects of changing requirements. However indications from elevated $DI(n)$ values as well as empirical evidence cited above suggest significant changes in the code between I_1 and I_2. How do we reconcile this contradiction?

- It is important to note there is a large body of *common* components across all of Browse requirements. This group contributes heavily in increasing the dependencies between the requirements and pushing up the respective $DI(n)$ values. These *common* components occur in more than 61% (11 out of 18) of the changes to existing components and more than 70% (29 out of 41) of the new components introduced, between I_1 and I_2. This in effect destroys much of the modularity of the underlying design, where small sets of independent components service each requirement.

- One of the vital insights we have gained into the workings of systems with changing requirements may be expressed as (paraphrasing the enduring motto of George Orwell's *Animal Farm*): *some requirements are more equal than others.* This boils down to the fact that every system will have requirements which are more

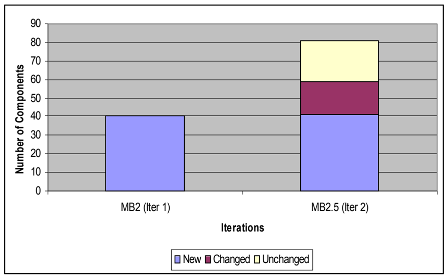

Figure 3: Variation of the number of code components for Browse between Morphbank 2.0 and 2.5

used by users and subject to greater changes, compared to other more stable ones. The $DI(n)$ values for these "more equal" requirements have to be as low as possible, such that no matter how high their $MI(n)$ values are for a particular iteration, the changes can be absorbed with minimal impact. So every requirement with same $DI(n)$ value indicates an uniformity of design that affects the system's ability to respond to changing requirements without much rework. It is expedient to implement requirements that change most in a way their components are the most loosely coupled in the system, with other less volatile requirements being serviced by more closely meshed components. These design tradeoffs are guided by the metrics. An optimal distribution of responsibility across the components will facilitate maximum responsiveness to changing requirements with minimal overall impact.

10. RECOMMENDATIONS

In view of the above discussion, we recommend the following:

- Given that Browse functionality is likely to undergo frequent changes in the future (for instance, there is likely to be a requirement to provide a different taxonomic structure to search which is not provided through ITIS), we suggest the Morphbank design be modified to reflect clearer separation of concerns across components. *Display*, *Processing*, and *Storage* aspects of a requirement's fulfillment should be implemented by separate, interacting components instead of ones doing all of these by themselves. This will ensure when a changing requirement affects one aspect, there is higher localization of corresponding code changes: if only the user interface changes there will be no need to modify components which also have database access logic in them, and so on.

- As Morphbank's services are preeminently web based, a Web-Service based architecture may offer better scalability. This will entail more intense development effort in the short run, which will be offset by the long term benefits in enhancement and maintenance.

- In the Browse functionality of Morphbank 2.0 and Morphbank 2.5 there is very little of what is called "business logic". However as the scope of the system is expanded in the future it is not unlikely there will be a need for more processing between the access and display of information. So introducing to a Model-View-Controller (MVC) pattern of architecture will be helpful.

11. FUTURE WORK

We plan to continue the case study into future versions of Morphbank. As further requirement changes need to be incorporated into the system, the metrics will be able to facilitate design decisions to absorb their effects with minimal impact.

We also plan to develop an Eclipse [7] based tool to automate the calculation and interpretation of the metrics. We expect to use the Morphbank case study to test and refine the tool.

12. CONCLUSION

In this paper we reported the results from a case study of applying a set of metrics to gauge the effects of changing requirements on the design of a system. The metrics, *Mutation Index*, *Component Set*, and *Dependency Index* [1], were applied on the Browse functionality of Morphbank – a web-based interactive system for biological research – across two of its releases, versions 2.0 and 2.5. The indications from the metrics on the extent of code change due to changing requirements were shown to correlate closely with the empirical data. Based on the observations, we also recommended certain modifications in the system's architecture to better absorb future requirement changes. We also mention future directions of our work.

13. REFERENCES

[1] S. Datta and R. van Engelen. Effects of Changing Requirements: A Tracking Mechanism for the Analysis Workflow. *Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC-SE-06) vol. 2, pp.1739–1744*, April, 2006.

[2] Morphbank. *http://www.morphbank.net/*, 2006.

[3] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.

[4] I. Jacobson,G. Booch,J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[5] Morphbank User Manual. *http://morphbank.net/docs/mbUserManual.pdf*, 2006.

[6] http://www.itis.usda.gov/. *http://www.itis.usda.gov/*, 2006.

[7] Eclipse - an open development platform. *http://www.eclipse.org/*, 2006.

[8] ExamDiff - Visual File Comparison Tool. *http://www.prestosoft.com/ps.asp?page=edp_examdiff*, 2006.