



# COMPUTING SCIENCE

Using Early Stage Project Data to Predict Change-Proneness

Claire Ingram and Steve Riddle

**TECHNICAL REPORT SERIES**

---

No. CS-TR-1321

March 2012

## **Using Early Stage Project Data to Predict Change-Proneness**

**C. Ingram and S. Riddle**

### **Abstract**

Several previous studies have suggested methods for predicting change-proneness based on software complexity metrics. We hypothesise that data from the early stages of a development project such as requirements and design could also be used to make such predictions. We define here a set of new metrics to capture data from the requirements and/or design stages, and derive values for these metrics using a case study project. We do find that significant differences in change-proneness can be detected between components with high or with low values for our metrics, suggesting that this is an area which would benefit from further study.

## Bibliographical details

**INGRAM, C., RIDDLE, S.**

Using Early Stage Project Data to Predict Change-Proneness

[By] C. Ingram, S. Riddle

Newcastle upon Tyne: Newcastle University: Computing Science, 2012.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1321)

### Added entries

NEWCASTLE UNIVERSITY

Computing Science. Technical Report Series. CS-TR-1321

### Abstract

Several previous studies have suggested methods for predicting change-proneness based on software complexity metrics. We hypothesise that data from the early stages of a development project such as requirements and design could also be used to make such predictions. We define here a set of new metrics to capture data from the requirements and/or design stages, and derive values for these metrics using a case study project. We do find that significant differences in change-proneness can be detected between components with high or with low values for our metrics, suggesting that this is an area which would benefit from further study.

### About the authors

Claire Ingram obtained a BA from the University of Durham, followed later by an MSc in Computing Science from Newcastle University. After graduating with an MSc she spent five years working in the north east of England in various industries as a software developer and website manager. She returned to Newcastle in 2007 to start a PhD, working under the supervision of Steve Riddle. After obtaining her PhD in 2011 she has continued work at Newcastle as a Teaching Fellow as well as researching prediction models in software engineering.

Dr Steve Riddle delivers courses in formal specification and software development techniques, software engineering and high-integrity software development. He also coordinates the undergraduate placement programme. He is currently a member of the EU FP7 Integrated Project COMPASS, which is undertaking research on model-based techniques for developing and maintaining Systems of Systems (SoS). Dr Riddle obtained his BSc in Computer Software Technology at the University of Bath in 1991, and completed a PhD at Bath in 1997 on the use of partial specifications and refinement theory to aid the process of explaining complex systems. His research interests include requirements engineering and traceability, software volatility, formal specification and evidence-based argumentation.

### Suggested keywords

CHANGE-PRONENESS

EARLY STAGE PROJECT

DESIGN

REQUIREMENTS

VOLATILITY

PREDICTION

# Using Early Stage Project Data to Predict Change-Proneness

Claire Ingram  
School of Computer Science  
Newcastle University  
Newcastle upon Tyne, NE1 7RU, UK  
claire.ingram@ncl.ac.uk

Steve Riddle  
School of Computer Science  
Newcastle University  
Newcastle upon Tyne, NE1 7RU, UK  
steve.riddle@ncl.ac.uk

**Abstract**—Several previous studies have suggested methods for predicting change-proneness based on software complexity metrics. We hypothesise that data from the early stages of a development project such as requirements and design could also be used to make such predictions. We define here a set of new metrics to capture data from the requirements and/or design stages, and derive values for these metrics using a case study project. We do find that significant differences in change-proneness can be detected between components with high or with low values for our metrics, suggesting that this is an area which would benefit from further study.

## I. INTRODUCTION

Some existing studies use complexity measures such as those proposed by [5] or [10] for predicting which components will become change-prone. We hypothesise that data from requirements and design activities may also prove to be useful in predicting change-proneness. In this paper we describe a feasibility study undertaken to test the validity of our hypothesis. To achieve this we firstly develop metrics for ‘quantifying’ requirements and design activities, generate values for these metrics from a real-world case study and finally compare our metrics with the actual number of changes detected. The rest of this paper is laid out as follows. Section II summarises previous work on predicting change-proneness and Section III explains our underlying hypotheses. Section IV describes how we model the early stages of a project and Section IV-A defines some new metrics to assess early stage project activity. Section V presents our case study project. Section VI presents our results and finally Section VII our conclusions.

## II. PREVIOUS STUDIES

Wilkie and Kitchenham [13] conducted a study on complexity metrics and change-proneness, concluding that the Coupling Between Objects (CBO) metric (proposed by [4]) could help to identify change-prone classes, whilst Briand *et al* [2] found a link between coupling measures and classes which change together. Chaumon *et al* [3] found that Weighted Methods per Class (WMC) (also proposed by [4]) could be used to predict classes propagating changes. Li and Henry [10] discovered correlations between a selection of code metrics and changes, although van Kotten and Gray

[12] concluded that statistical models which reused this data were not very accurate. Relatively little work examines the use of data from an early stage of the project, although Jiang *et al* found that requirements metrics improved a metrics-based fault prediction model [9].

## III. USING REQUIREMENTS AND DESIGN DATA

We do not impose value judgements on change-proneness or try to eradicate volatility, but hope that our approach allows planners to minimise the ill effects of unexpected changes.

Our underlying hypothesis is that activity during the requirements and/or design phases influences later change-proneness of individual components (although other factors play a substantial role as well). Components linked to many requirements, for example, may be *more* likely to change later because requirements changes propagate ‘ripples’. Alternatively, a component linked to many requirements may occupy a difficult-to-alter intersection between functions, insulating it from change requests. Similarly, if there are many design decisions documented for a component, it could be a sign that it is controversial or ‘difficult’. If many options for a decision have been documented, designers may have been struggling to find a solution, and this could mean more change requests later, when some stakeholders find themselves disappointed. Alternatively, if a component is linked to many decisions or options it could be a sign that it benefitted from a more detailed investigation or an extended dialogue with users, leading to fewer unforeseen incompatibilities and misunderstandings, and fewer subsequent changes.

Many design decisions will be made without being documented, because they do not arise until mid-implementation, or are made implicitly, or are not regarded as significant enough to record. Only those decisions and options which the designer considers significant enough to write down are used as information sources in our study. We believe that this ‘filter’ effect is useful, as decisions that were important enough to document are likely to have non-negligible effect on development.

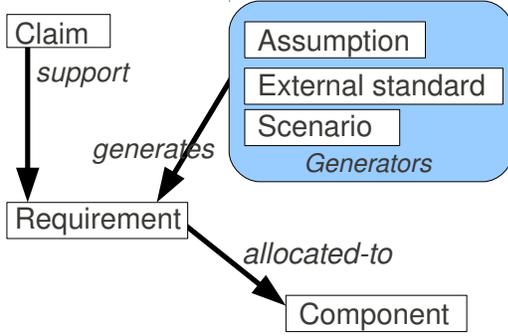


Figure 1: Proposed model of entities and relationships for the requirements phase

#### IV. DEVELOPING NEW METRICS

In this section we describe how we developed some metrics for ‘measuring’ the extent of these requirements and design activities. We follow the model-order-mapping (MOM) approach [6] for developing new metrics, in which a model is created to represent a real-world document (such as a code module) and its characteristics of interest. Mappings are defined between the model and the real-world document, and between the model and an ‘answer set’ of ‘real numbers’ [6]. To create our models, we identified some key concepts from the requirements and design stages that may help predict future changes. We represent requirements themselves and also some rationale, which may take the form of assumptions, external standards or user scenarios (collectively termed *Generators*). The ‘user scenario’ could be a use case, or an interview with a stakeholder that records what users expect. User expectations are an important motivator for change so we are keen to include some ‘measure’ of user input. The ‘Claim’ is a simple statement used to justify the inclusion of a requirement (e.g., *including this function will allow us to gain a competitive advantage by...*). We are also interested in External Standards, which we define as any standard published by a third party. These tend to be very stable, and components implementing them may thus be somewhat insulated from change. Alternatively, repeated refinements may be needed to balance the dual goals of adherence to a standard whilst satisfying local requirements, resulting in higher change-proneness. For the design stage we represent a series of decisions and their options. Decision outcomes are collectively termed *Outcomes* and can be an internal standard (a standard developed by the team themselves), a choice between options (e.g., *Out of all options considered, we selected to use x*), or a ‘constraint’, which is a statement about how some feature will be implemented. One decision may produce many constraints.

In our models, the entities we have proposed are linked by relationships, following a logical progression from requirements rationale towards requirements, issues, options

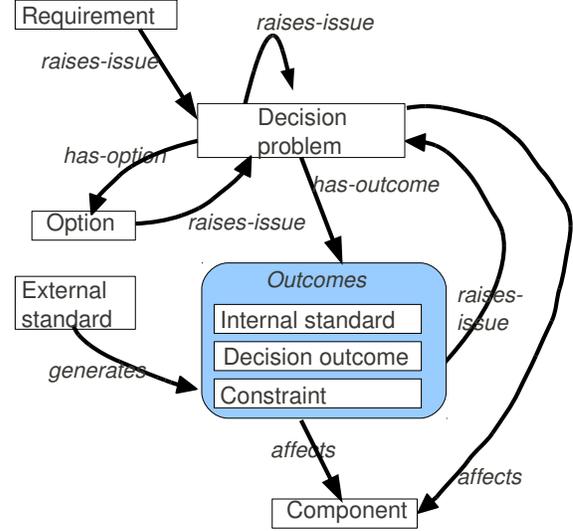


Figure 2: Proposed model of entities and relationships for the design phase

and design decisions, and towards components. Figures 1 and 2 show the proposed structures.

##### A. Developing Metrics from the Models

We now define a series of metrics to measure attributes of our models. Definitions are provided in Table I, in mathematical set building notation. We represent both the system and entities we have listed above (such as requirements, assumptions, etc) as a directed graph,  $G$ , which contains entities  $(x, y, z \dots)$ . Edges connect those entities to indicate the presence of a relationship. We write  $(x, y, l)$  to refer to entities  $x$  and  $y$  linked by relationship with label  $l$ , or  $(x, y)$  to indicate a linked pair of entities with any label. The set of entities  $G$  contains several meaningful subsets of entities:

- *Reqs* is the set of all Requirements.
- *Cmpis* is the set of all Components.
- *Optns* is the set of all Options.
- *Claims* is the set of all Claims.
- *Genrs* is the set of all Scenarios and Assumptions and External Standards.
- *DP* is the set of all Decision Problems.
- *Outcomes* is the set of all Internal Standards and Decision Outcomes and Constraints.
- *Externals* is the set of all External Standards.

We write  $M(c)$  to refer to a metric  $M$  calculated for component  $c$ . Where duplicate relationships exist between  $(x,y)$  we count only one, following the standard semantics of a set. This makes the models simpler to implement, and allows us to reason about the sets in a standardised way. In practice we did not find a large number of such duplicates.

- *Number of Requirements (NR)*.  $NR(c)$  is the number of requirements which have been allocated to a component

Table I: Metric definitions

$\mathbf{NR}(c) =  \{x \mid (x, y) \in G : y = c \wedge x \in Reqs\} $ where $Reqs$ is the set of requirements	
$\mathbf{SRR}(c) = ReqsGenerators(c) + ReqsClaims(c)$ where $Reqs(c) = \{x \mid (x, c) \in G : x \in Reqs\}$ and $Reqs$ is the set of all Requirements and $ReqsGenerators(c) =  \{x \mid (x, y, l) \in G : y \in Reqs(c) \wedge x \in Genrs \wedge l = \text{'generates'}\} $ and $ReqsClaims(c) =  \{x \mid (x, y, l) \in G : y \in Reqs(c) \wedge x \in Claims \wedge l = \text{'supports'}\} $	
$\mathbf{NOut}(c) =  \{x \mid (x, y) \in G : y = c \wedge x \in Reqs \wedge x \in Outcomes\} $	$\mathbf{ASRR}(c) = \frac{SRR(c)}{NR(c)}$
$\mathbf{NES}(c) = ReqsExternals(c) + OutcomeExternals(c)$ where $ReqsExternals(C) =  \{x \mid (x, y, l) \in G : y \in Outcomes(c) \wedge x \in Externals \wedge l = \text{'generates'}\} $ and $OutcomeExternals(C) =  \{x \mid (x, y, l) \in G : y \in Requirements(c) \wedge x \in Externals \wedge l = \text{'generates'}\} $	
$\mathbf{NDP}(c) =  \{x \mid (x, y, l) \in G : y \in Cmpnts \wedge x \in DP \wedge l = \text{'affects'}\} $ $+  \{x \mid (x, y, l) \in G : y \in Outcomes(c) \wedge x \in DP \wedge l = \text{'has-outcome'}\} $ where $Outcomes(c)$ counts Outcomes linked to $c$	
$\mathbf{NOpt}(c) =  \{x \mid (x, y, l) \in G : y \in DP(c) \wedge x \in Optns \wedge l = \text{'has-option'}\} $ where $DP(c)$ are Decision Problems linked to $c$ (see $NDP(c)$ ).	
$\mathbf{ANODP}(c) = \frac{NOpt(c)}{NDP(c)}$ where both $NOpt(c)$ and $NDP(c)$ have been defined earlier.	

$c$  (definition provided in Table I). In other words, all those relationships  $(x,y)$  in the graph  $G$  which connect component  $c$  with any entity from the set of Requirements.

- *Strength of Requirements Rationale (SRR)*. SRR is intended to measure recorded rationale. If  $r$  is a requirement linked to  $c$ ,  $SRR(c)$  is the total sum of Generators linked to  $r$  plus the Claims that support  $r$ .
- *Average Strength of Requirements Rationale (ASRR)*. A mean figure of rationale recorded per requirement.
- *Number of Outcomes (NOut)*. Counts Decision Outcomes affecting component  $c$ .
- *Number of External Standards (NES)*. This metric quantifies external standards associated with a component, via Requirements or Decision Outcomes.
- *Number of Decision Problems (NDP)*. Counts Decision Problems linked to components via the Outcomes or directly.
- *Number of Options (NOpt)*. The number of Options that can be linked to a Component via Decision Problems.
- *Average Number of Options per Decision Problem (ANODP)*. The total number of Options which can be linked to a component, divided by the total number of Decision Problems linked to the same component.

A framework for validating software metrics proposed by Briand *et al* [1] distinguishes types of metric, and can be applied to non-code entities like ours. We believe that NR, NOut, NDP, SRR, NA, NES and NOpt satisfy Briand *et al*'s notion of a size metric; this is logical as we aim to express *quantity* of requirements or design activity. NR, NES and NOpt also satisfy Briand *et al*'s notion of complexity metrics. However, ANODP and ASRR are averages and do not satisfy any of the properties, so we proceed with caution for these metrics.

## V. CASE STUDY PROJECT

Our investigation uses CARMEN (Code Analysis Repository & Modelling for e-Neuroscience) as a case study

(described in an earlier paper [8]). This is an e-Science Pilot Project funded by the Engineering and Physical Sciences Research Council (EPSRC) (UK), which aims to create a 'virtual laboratory for neurophysiology'<sup>1</sup>. We focus on the work of the team handling Work Package 0 (WPO), who implement the platform and data-sharing capabilities, with a team of 7 developers at its maximum size. The platform is written in Java, with a distributed architecture. Initially WPO adopted a waterfall-style process model, but this has been gradually restructured into a series of iterations over a spiral-style model. CARMEN's development team use Subversion<sup>2</sup> and data was extracted from Subversion's logs to generate a list of all Java class files comprising the system, a total of 254 unique files. We assumed that a single Java class file mapped to one of our components (nested classes are not counted separately). We could have used another mapping (e.g., packages instead of individual classes) but using classes gives us the right amount of granularity for our study in a system the size of CARMEN.

We imported all of the requirements from WPO's documentation, a total of 152. CARMEN's team do not maintain a documented mapping to connect requirements to components, so we examined each component and manually linked requirements and components (steps were taken to validate these links, summarised in Section VI-A). Design-related entities (such as Decision Problems, Options, Outcomes etc.) were populated by extracting data from the design documentation, requirements, interviews with users, technical reports and specification documents. The design document narrowed down the wide field of possible choices to a specific path; these statements were represented in our structures as Constraints. The presence of a Constraint indicates that a decision must have been taken, and a Decision Problem was added, along with any documented options. After completing the above stages, the total number of entities, including components, design entities and requirements, is 852, with

<sup>1</sup><http://www.carmen.org.uk/about>

<sup>2</sup><http://subversion.apache.org/>

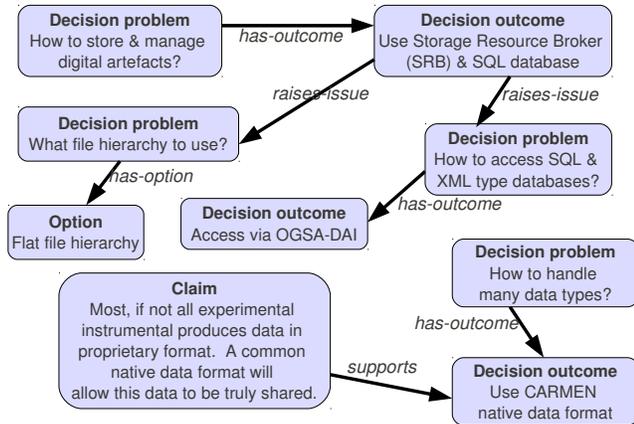


Figure 3: An example of design entities created using data from CARMEN System Architecture Document

2095 relationships linking these entities. As part of the process of populating our models, we randomly selected 10% of entities from our database with their surrounding structure and showed them to developers from CARMEN, to ensure continued accuracy. Figure 3 presents an example of a populated structure from the case study.

To measure change-proneness, we could count the number of times a component/file has been checked in to Subversion, or we could count the lines changed (as in [10], [12]) between check-in events and/or fixed dates. Either can be skewed by individual developer habits, since some developers may check in work every night while others prefer to complete work before committing. For this case study, we use frequency of check-in events as a surrogate for number of changes made. We discount the first, initial check-in. Deletions are counted as a single change.

## VI. RESULTS

Analyses are carried out using Minitab<sup>3</sup>. Our data could not be matched to any well-known data distributions, and we failed to uncover a suitable transformation. Thus for our analyses we employ non-parametric tests which do not make assumptions about underlying distribution. All the tests we use generate probabilities indicating test significance. Usually this is the probability that we will commit the error of falsely rejecting the null hypothesis  $H_0$ .  $\alpha$  indicates the threshold at which we are prepared to reject  $H_0$ . We set  $\alpha$  to 0.05, meaning that the chance of making this error in any of our tests is 5% or less. We used the non-parametric Kruskal-Wallis and Mann-Whitney<sup>4</sup> tests. These tests involve dividing the components into categories (or ‘bins’) based on the component’s value of metric  $m$ , and examine change-proneness experienced by components in

<sup>3</sup><http://www.minitab.com/>

<sup>4</sup>This test may be called the Wilcoxon rank sum test, Mann-Whitney U test, Mann-Whitney-Wilcoxon (MWW), or Wilcoxon-Mann-Whitney.

Table II: Summary Results of the Kruskal-Wallis tests on our metrics

$p$ (adj)	H	Groups	$n$	Median
<b>NR</b>				
0.000	64.17	NR=0	57	1.0
		NR=1	28	3.0
		NR=2	35	2.0
		NR=3	29	5.0
		NR=4	20	7.5
		NR=5	19	12.0
		NR=6	9	8.0
		NR=7	10	7.5
		NR>7	47	14.0
<b>NOut</b>				
0.000	66.63	NOut=0	73	1.0
		NOut=1	103	4.0
		NOut=2	19	12.0
		NOut=3	13	9.0
		NOut=4	12	11.0
		NOut>4	34	11.5
<b>NOpt</b>				
0.000	51.61	NOpt=0	91	1.0
		1<=NOpt<3	105	7.0
		3<=NOpt<5	15	8.0
		5<=NOpt<7	6	11.5
		7<=NOpt<9	21	8.0
		NOpt>9	16	12.5
<b>NDP</b>				
0.000	69.40	NDP=0	65	1.0
		NDP=1	29	1.0
		NDP=2	79	7.0
		NDP=3	16	10.0
		NDP=4	11	8.0
		NDP=5	13	9.0
		NDP=6	15	12.0
		NDP>6	26	11.0
<b>ANODP</b>				
0.000	51.93	ANODP=0	91	1.0
		0<ANODP<1	46	12.0
		ANODP=1	79	7.0
		1<ANODP<2	26	8.5
<b>SRR</b>				
0.000	64.36	SRR=0	85	1.0
		SRR=1	9	1.0
		SRR=2	71	4.0
		SRR=3	24	8.0
		SRR>3	65	12.0
<b>ASRR</b>				
0.000	66.42	ASRR=0	85	1.0
		0<ASRR<=0.5	35	9.0
		0.5<ASRR<1.0	43	9.0
		ASRR=1	36	2.0
		1<ASRR<=1.5	22	13.0
		1.5<ASRR<=2.0	27	4.0
		ASRR>2.0	6	4.5

each category, to see if there is a difference between categories. The null hypothesis for both of these tests is that data in all groups is drawn from samples with identical medians for change-proneness. The Mann-Whitney test compares results for two categories of observations; it produces a statistic  $W$  which tends to be greater if the first sample contains larger values, and a probability  $p$  that two groups are observed with values as separated as these when the populations in fact have the same median. The

Kruskal-Wallis test is very similar to the Mann-Whitney test, but generalised to compare many categories in one test. However, it can only indicate that a difference exists between two of the many groups in the test; it does not indicate which two categories these are. To identify *which* two groups differ [11], we pair up each category for metric  $m$  with every other category for  $m$ , and execute a Mann-Whitney test on each pairing. In this case, if  $\alpha$  is 0.05 for *each* individual pairing, then our overall potential for error can become unacceptably large, since there will be many unique pairings within one test. The Bonferroni adjustment is one well-known technique for controlling  $\alpha$  across many tests; this adjustment simply divides the desired  $\alpha$  by the number of paired comparisons  $k$ . This is very cautious, however, such that there is a risk some useful differences are missed [11]. Holm has proposed instead a sequentially rejective method [7], [11] for controlling  $\alpha$  across many linked tests, which we use in our analysis. Following the Holm technique, pair-wise comparisons 1 to  $k$  are ranked, with the most significant (i.e., with the lowest  $p$ ) ranked as 1.  $\alpha$  is then calculated individually for each pair-wise comparison as:  $\alpha = \frac{\alpha}{k - rank + 1}$ . For the first comparison,  $\alpha$  is therefore  $\alpha/k$ ; for the second,  $\alpha$  is  $\alpha/k - 1$ ; and so on. When a comparison is encountered that exceeds  $\alpha$ , that and all subsequent null hypotheses must be accepted.

For NES, there was only sufficient variety to support two groups:  $m=0$  and  $m>0$ , so we used a two-sample Mann-Whitney test. The test produced  $p = 0.000$ , suggesting that there is a statistically significant difference between the two categories. For each other metric  $m$ , we grouped components into many categories based on metric  $m$  and executed a Kruskal-Wallis test on all categories. When defining categories, we try to achieve as far as possible a series of regularly-spaced groups, each containing at least five components (results may be unreliable if categories contain fewer than this). Results are summarised in Table II. Kruskal-Wallis tests for all metrics produce a  $p$  value below  $\alpha$ , showing that a difference between categories exists. Therefore we pair each category with all other categories and execute Mann-Whitney tests on each pairing, following the Holm technique. We use these Mann-Whitney tests to identify thresholds where values start to become significant. Some general observations we make as a result of analysis on all metrics are summarised in the list below. Components with:

- NES $\geq 0$  are more likely to be change-prone.
- NR $\geq 3$  are more likely to be change-prone.
- SRR $\geq 2$  are more likely to be change-prone than those with SRR=0 or SRR=1
- ASRR=0 or ASRR=1 are less likely to be change-prone, whilst  $0 < ASRR < 1$ , or  $1 < ASRR < 2$ , are more likely.

Table III: Number of components in the corrected sample (out of total 23) which saw changes in metric values after developer corrections

	NR	SRR	ASRR	NOut
<b>No. changed</b>	4	1	2	2
<b>% changed</b>	17.39%	4.35%	8.7%	8.7%
	NDP	NOpt	ANODP	NES
<b>No. changed</b>	2	1	2	0
<b>% changed</b>	8.7%	4.35%	8.7%	0%

Table IV: Results of Mann-Whitney tests on the validated sample

Metric	N	median	$p$ (adj)	95% Confidence Interval	W
NR low & NR high	9 14	1.0 11.0	0.0032	(-15.00,-2.00)	61.0
SRR low & SRR high	7 16	2.0 11.0	0.0224	(-15.00,-1.00)	49.5
ASRR low & ASRR high	10 13	1.5 12.0	0.0018	(-16.00,-3.00)	69.5
NOut low & NOut high	17 6	2.0 14.5	0.0243	(-21.004,-1.00)	171.5
NDP low & NDP high	9 14	1.0 12.0	0.0008	(-16.00,-2.00)	54.5
NOpt low & NOpt high	9 14	1.0 12.0	0.0008	(-16.00,-2.00)	54.5
ANODP low & ANODP high	10 13	1.0 12.0	0.0041	(-16.00,-1.00)	73.5

- NOut $\geq 2$  are more likely to be change-prone than components with NOut $< 2$ .
- NDP $\geq 2$  are more likely to be change-prone.
- NOpt $> 0$  are more likely to be change-prone.
- $0 < ANODP < 2$  are more likely to be change-prone than components with ANODP=0 or ANODP $> 2$ .

To validate the models we have constructed of CARMEN, we randomly selected 10% of the components (25 entities), listed all the requirements and design issues which had been allocated to each, and asked CARMEN developers to check that they agreed with the assignments. Any corrections they suggested were made to the sample. Two files in our random sample had been imported early on by a team member who had subsequently left, so the remaining team were unable to comment on them and we were forced to exclude them from the sample. After changes had been implemented, metric values were regenerated for the 23 corrected components. A few metrics values for components in the corrected sample had changed as a result. Table III shows how many components in the sample had metrics values which changed after validation.

For almost all metrics 90% of components in the sample did not change metric value when corrected by developers; in all cases more than 80% were unchanged. We assume that this suggests system developers generally agree with our generated metric values.

### A. Validating Requirements Links

For each metric we divided the validated sample of 23 components into two groups, labelled ‘low’ and ‘high’, by following the observations made in Section VI. For example, we had observed that components with NOut of 2 or more are more likely to become change-prone, so we place components in the corrected sample with NOut of 2 or more into a group labelled ‘NOut-high’, and all other components into a ‘NOut-low’ group. NES is excluded as there are insufficient components to create two groups. We then executed two-sample Mann-Whitney tests against the groups; results are summarised in Table IV.

Our results suggest that statistically significant differences in change-proneness can be detected for components with lower and higher values of NR, SRR, ASRR, NOut, NDP, NOpt and ANODP in our validated sample. Since NES did not see any changes in values in our corrected sample we assume that observations made earlier still stand. For all of these metrics, then, we assume that the system-wide predictions (i.e., on an uncorrected sample) are not contradicted by the results of the corrected sample. On the other hand, we have insufficient data in the corrected sample to verify that our observations on ASRR still stand; our conclusions on this metric therefore remain very tentative.

### B. Generalisability

The capturing and structuring of requirements and design data clearly varies enormously, but we believe our approach could be generalised to any project that has requirements and/or design information available (although further studies are needed to confirm results). However, our results are unlikely to be applicable to projects adopting a very different approach to requirements and/or design (e.g., an agile development).

Our case study includes changes that have been planned in advance, and are not unexpected. However, over a period of several years, the system does evolve. For example, out of the 254 components which were included in our study, 145 had been included in the project and then removed some time before we extracted details from subversion (our study covers approximately 4 years of the project). This suggests that many of the changes made were not ones which were expected at the outset. Our approach can easily be extended to components which are newly added to a project, by modelling links between existing design/requirements to the new component. We assume that most projects covering a number of years will undertake refactoring at some point, and will see a similar evolution of the system.

### C. Evaluating Success of our Predictions

To evaluate our results, we define a region of volatile components in which we are interested. Figure 4 is a histogram showing frequency of numbers of changes experienced by components in CARMEN. There are a number

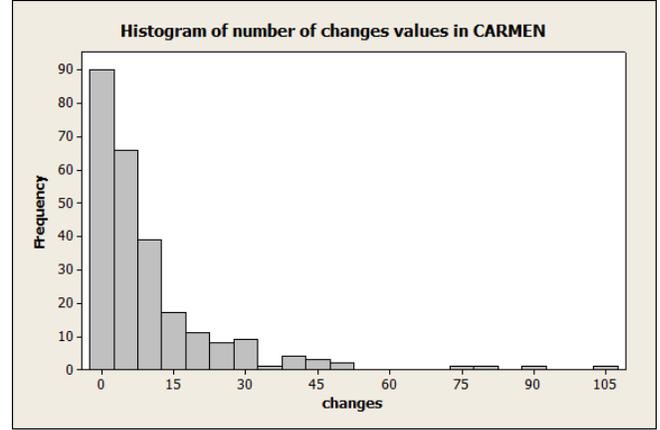


Figure 4: Histogram of number of changes

Table V: Probability that any component in the system may fall into one of our change-prone tiers (assuming all components have an equal chance of changing)

Volatile group	probability $p$ a component belongs to it
top 1.5% ( $\geq 60$ changes)	0.0157
top 7% ( $\geq 30$ changes)	0.0827
top 13% ( $\geq 20$ changes)	0.1299

of points where we could set a threshold to delimit ‘volatile’ components of interest:

- We should certainly like to identify the 4 outliers with numbers of changes higher than 60. This represents the top 1.5% most volatile components.
- We could place a threshold at 30. This represents the top 7% of most volatile entities - 18 components. Figure 4 shows that the number of components drops sharply past 30 changes.
- We could place a threshold at 20. This represents the top 13% of volatile entities (33 components). The frequency of number of changes also drops as it falls between 20.

For our evaluation we use all three thresholds and evaluate metrics’ ability to predict the contents of all three tiers. We choose these points because:

- Most components experience fewer changes than this - these are the extremes
- These thresholds represent values where the frequency of components falls naturally, meaning that there is a clear difference between numbers of components with changes higher than this and lower than this.
- The top 1.5%, top 7% and top 13% components are reasonably spaced intervals.
- Different types of projects may have different objectives for the total number of volatile components they wish to predict, or where they would set their threshold.

Table VI: Probability  $p$  that a component will be successfully predicted change-prone by each metric, compared to the overall probability for any component for the top 1.5% most volatile components

Metric value	No. predicted in group	Prob. that cmpts are change-prone	Diff. between the two groups	Diff. btwn predicted group & overall $p$
<b>top 1.5% most volatile components (&gt;60 changes)</b> Overall probability: 0.0157				
NR<3	119	0.0084		
NR>=3	135	0.0222	+1.38%	+0.65%
NDP<2	166	0.0		
NDP>=2	88	0.0455	+4.545%	+2.98%
SRR<=2	166	0.012		
SRR>2	88	0.0227	+1.068%	+0.70%
ASRR=1, ASSR=0, ASRR>=2	142	0.007		
0<ASRR<1, 1<ASRR<2	112	0.0268	+1.97%	+1.11%
NOut<2	176	0.0114		
NOut>=2	78	0.0256	+1.43%	+0.99%
NES=0	240	0.0.125		
NES>0	14	0.0714	+5.89%	+5.57%
NOpt=0	91	0.0		
NOpt>0	163	0.245	+2.45%	+0.88%
ANODP=1, ANODP>=2	91	0.0		
0<ANODP<2	163	0.02	+2.0%	+0.43%

We calculate the overall probability  $p$  that any of the 254 components in the system will fall into one of these ‘change-prone’ groups (assuming that each component has an equal chance of being changed); figures are shown in Table V. Next, for each metric  $m$ , we divide the population of components into two groups: one which is predicted by  $m$  to be change-prone, according to the rules we devised in Section VI, and a second group which is not predicted by  $m$  to be change-prone. So, for example, we stated earlier that we had observed that components with  $NR>3$  are more likely to be change-prone, so for the metric  $NR$  we separate components into groups:  $NR \geq 3$  and  $NR < 3$ . Then we calculate the probability of a component in each of the two groups falling into one of the ‘change-prone’ groups or not. The results are shown in Tables VI to VIII. We also include in these tables the total number which  $m$  predicted would be change-prone, since a metric which ‘predicts’ change-proneness for too many components is less likely to be useful.

Table VII: Probability  $p$  that a component will be successfully predicted change-prone by each metric, compared to the overall probability for any component for the top 7% most volatile components

Metric value	No. predicted in group	Prob. that cmpts are change-prone	Diff. between the two groups	Diff. btwn predicted group & overall $p$
<b>top 7% most volatile components (&gt;30 changes)</b> Overall probability: 0.0827				
NR<3	119	0.0336		
NR>=3	135	0.1259	+9.23%	+4.32%
NDP<2	166	0.006		
NDP>=2	88	0.2273	+22.13%	+14.46%
SRR<=2	166	0.0301		
SRR>2	88	0.1818	+15.17%	+9.91%
ASRR=1, ASSR=0, ASRR>=2	142	0.0352		
0<ASRR<1, 1<ASRR<2	112	0.1429	+10.77%	+6.02%
NOut<2	176	0.017		
NOut>=2	78	0.2308	+21.37%	+14.81%
NES=0	240	0.0625		
NES>0	14	0.4286	+36.61%	+34.59%
NOpt=0	91	0.011		
NOpt>0	163	0.1227	+11.17%	+4.0%
ANODP=0, ANODP>=2	91	0.022		
0<ANODP<2	163	0.117	+9.46%	+3.37%

Table VIII: Probability  $p$  that a component will be successfully predicted change-prone by each metric, compared to the overall probability for any component: top 13% most volatile components

Metric value	No. predicted in group	Prob. that cmpts are change-prone	Diff. between the two groups	Diff. btwn predicted group & overall $p$
<b>top 13% most volatile components (&gt;20 changes)</b> Overall probability: 0.1299				
NR<3	119	0.042		
NR>=3	135	0.2074	+16.54%	+7.75%
NDP<2	166	0.012		
NDP>=2	88	0.3523	+35.022%	+22.24%
SRR<=2	166	0.0542		
SRR>2	88	0.2727	+21.851%	+14.28%
ASRR=1, ASSR=0, ASRR>=2	142	0.0563		
0<ASRR<1, 1<ASRR<2	112	0.2232	+16.688%	+9.33%
NOut<2	176	0.0568		
NOut>=2	78	0.2949	+23.81%	+16.5%
NES=0	240	0.1042		
NES>0	14	0.5714	+46.73%	+44.15%
NOpt=0	91	0.033		
NOpt>0	163	0.184	+15.11%	+5.41%
ANODP=0, ANODP>=2	91	0.044		
0<ANODP<2	163	0.178	+13.34%	+4.81%

We find that the 4 most volatile components (the top 1.5%) are very difficult to predict. We have calculated the probability of any single component being change-prone for each tier of change-proneness, knowing nothing about the component and assuming that all components have an equal chance of being change-prone. In the top 1.5% tier our metrics only outperform this measure by very small percentages. This suggests that these four, very volatile components are perhaps influenced by many other factors. We have more success with other volatile components.

## VII. CONCLUSIONS AND FURTHER WORK

We found some evidence that some metrics can be used to differentiate between groups of components in terms of change-proneness. In order to show the sensitivity of our metrics we have calculated the different probabilities that components selected (or not) by a metric will be change-prone, and the difference between these two. Our metrics do ‘select’ components with a higher probability of change than the ‘non-predicted’ group, and also with a higher probability of change than exists overall for all components. NES achieves a particularly strong performance here, improving on the overall population-wide probability of change-proneness by 44.15% for the top 13% most volatile components. Many of our metrics also make quite large predictions, however, which must reduce their sensitivity as possible predictors.

The exception to this is the NES metric, which is parsimonious as well as successfully predicting some change-prone components. Our results appear to suggest that links to external standards can make a component more change-prone, as shown by results for SRR, ASRR and NES. A possible explanation is that iterative changes are initially needed to ensure dual goals of adherence to a standard and satisfaction of local requirements are achieved. We might expect that the actual effects of dependencies on external standards are only detectable over a long period of time, however. Our current case study cannot supply years of maintenance data yet to confirm this.

Many of our metrics are potential indicators of the quantity of design effort. Our results for NOut, NDP and NOpt generally imply a higher probability of change-proneness associated with higher values of design metrics, which could suggest that the metrics can indicate components that were controversial or difficult to design, which has an effect on later changes. Despite these observations, we have noticed that the four *most* volatile components sit outside the very highest value groups of ANODP, NOpt, NDP and NOut. These can be considered outliers; the least volatile of this four is considerably more volatile than the next most changeable component. It could be that these extremely volatile components are affected by other factors and would be difficult to predict in any case. Nevertheless, we believe that

this exploratory study has shown that some measure of design effort and/or requirements effort can feasibly be linked to future change-proneness for many components. Further work is now underway to evaluate the effectiveness of this approach with a larger project and to test whether the same increases in probability of change-proneness can be seen in other development projects, or whether a combination of metrics can be employed to improve prediction quality. An area of uncertainty in our current approach is how real-world project data should be mapped to our models (e.g., what is a ‘decision’, what is an ‘outcome’?) Therefore future work aims to clarify the process of populating our models as well as testing robustness and generalisability of our conclusions.

## ACKNOWLEDGMENT

The authors would like to thank the CARMEN project staff for providing access to project data and team members. This work is supported by a CASE studentship awarded by BAESYSTEMS and the U.K. Engineering and Physical Sciences Research Council.

## REFERENCES

- [1] Lionel Briand, Sandro Morasca, and Victor R. Basili. Property-based software engineering measurement. *IEEE Transactions on Software Engineering*, 22:68–86, 1996.
- [2] Lionel C. Briand, Jürgen Wüst, and Hakim Lounis. Using coupling measurement for impact analysis in object-oriented systems. In *Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on*, pages 475–482, 1999.
- [3] M. Ajmal Chaumon, Hind Kabaili, Rudolf K. Keller, François Lustman, Département Iro, and Université De Montréal. A change impact model for changeability assessment in object-oriented software systems. In *Proceedings of the Third Euromicro Working Conference on Software Maintenance and Reengineering*, pages 130–138, 1999.
- [4] S. Chidamber and C. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.
- [5] Shyam R. Chidamber, David P. Darcy, and Chris F. Kemerer. Managerial use of metrics for object-oriented software: An exploratory analysis. *IEEE Transactions on Software Engineering*, 24(8):629–639, 1998.
- [6] David A. Gustafson, Joo T. Tan, and Perla Weaver. Software measure specification. In *SIGSOFT FSE*, pages 163–168, 1993.
- [7] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.
- [8] Claire Ingram and Steve Riddle. Linking software design metrics to component change-proneness. In *WeTSOM 2011 - 2nd International Workshop on Emerging Trends in Software Metrics (WeTSOM 2011)*, Honolulu, Hawaii, USA, 2011.

- [9] Yue Jiang, Bojan Cukic, and Tim Menzies. Fault prediction using early lifecycle data. In *Proceedings of the 18th IEEE International Symposium on Software Reliability, ISSRE '07*, pages 237–246, Washington, DC, USA, 2007. IEEE Computer Society.
- [10] Wei Li and Sallie Henry. Object oriented metrics which predict maintainability. Technical report, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, February 1993.
- [11] Phil Scholfield. Simple and sophisticated Bonferroni adjustment. <http://privatewww.essex.ac.uk/~scholp/bonferroni.htm>, 2009. [Visited 25th June 2009].
- [12] C. van Koten and A.R. Gray. An application of Bayesian network for predicting object-oriented software maintainability. *Information and Software Technology*, 48(1):59–67, 2006.
- [13] F.G. Wilkie and B.A. Kitchenham. Coupling measures and change ripples in C++ application software. *Journal of Systems and Software*, 52(2-3):157–164, 2000.