More Success and Failure Factors in Software Reuse

Tim Menzies, Justin S. Di Stefano

I. INTRODUCTION

***Abstract*— Numerous discrepancies exist between expert opinion and empirical data reported in Morisio et.al.'s recent TSE article. The differences related to what factors encouraged successful reuse in software organizations. This note describes how those differences were detected and comments on their methodological implications.**

KEYWORDS: reuse, machine learning

In the April 2002 TSE article *Success and Failure Factors in Software Reuse* [1], Morisio et.al. sought key factors that predicted for successful software reuse. Their data came from a set of structured interviews conducted with project managers of 24 European projects from 19 companies in the period 1994 to 1997. Those projects were trying to achieve company-wide reuse of between one to a hundred assets. Nine of those 24 projects were judged by their respective managers as failures. Morisio et.al. employed a well-designed interview process to collect a wide set of project attributes (for a complete listing of those attributes, see the appendix).

There is much that is exemplary in the approach taken by Morisio et.al. For example, their data collection method is well-documented. Also, an extensive manual analysis of their data is presented in the paper, including a full discussion of all nine failing reuse projects. §6 of that paper "A Reuse Introduction Decision Sequence" offers a detailed set of recommendations for organizations seeking to create reusable assets. Their related work section takes care to contrast their results with other researchers. An appendix to the paper shows a clustering analysis of the projects and the decision tree of Figure 1 that Morisio et.al. argue represents the two major predictors for reuse.

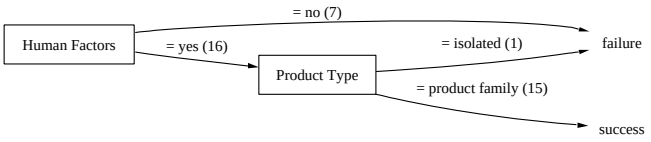

Fig. 1. A decision tree learnt by the CART data mining algorithm [5] from the Morisio et.al. data. Numbers denote how many examples exercise some edge.

Another exemplary feature of the Morisio et.al. study was that they presented their entire data set in their article. The inclusion of this data set allows other researchers to check their conclusions. When we checked their conclusions using several data miners, we found patterns that disagree with the decision sequence described in §6 of Morisio et.al. These differences are summarized in Figure 2.

Before focusing on those disagreements, it is important to stress that, in many aspects, we agree with Morisio et.al. For example, Figure 3 shows many attributes for which neither Morisio et.al. nor ourselves could find evidence that they predicted for successful reuse (see the entries marked with a "×"). For example, one of these "no evidence" attributes was use of $Development\ Approach = OO$. We completely endorse Morisio et.al.'s point that (e.g.) switching to C++ is insufficient to guarantee a successful reuse project. As to the other "no evidence" attributes", our studies don't say they don't matter: only that they did not appear to matter in the projects sampled by Morisio et.al. Figure 3 also shows other attributes that both our studies report predict for successful reuse. However, we could only find *barely supportive* or *very weak supportive* for some of those attributes (*barely supportive* and *very weak support* are defined below).

Attribute	Morisio et.al.	this paper
Application Domain	Not analyzed	×
Size of Baseline	Not analyzed	✓
Production Type	✓	×
Top Management Commitment	✓	✓ (barely)
Reuse Approach	×	✓
Domain Analysis	×	✓

Fig. 2. Conclusions where we disagree with Morisio et.al. ×/✓= no evidence/some evidence (respectively) in this data set that this attribute is relevant to determining success or failure of a reuse project. The label *barely* is explained in the text.

Attributes	Morisio et.al.	this paper
SP maturity	×	×
Software Staff	×	×
Overall Staff	×	×
Staff Experience	×	×
Type of Software	×	×
Development Approach	×	×
Software and Product	×	×
Origin	×	×
# assets	×	×
Qualification	×	×
Rewards Policy	×	×
Work Products	×	×
Independent Team	×	×
When Assets Built	×	×
Configuration Management	×	×
Key Reuse Roles Introduced	×	✓ (very weak)
Repository	✓	✓
Human Factors	✓	✓
Reuse Processes Introduced	✓	✓ (barely)
Non-Reuse Processes Modified	✓	✓ (very weak)

Fig. 3. Conclusions where we agree with Morisio et.al. ×/✓= no evidence/some evidence (respectively) in this data set that this attribute is relevant to determining success or failure of a reuse project. All the × marks in the middle column denote attributes which were not mentioned in §6 of Morisio et.al. and were not in the decision tree they learnt from their data (see Figure 1). All the ✓ marks in the right-hand column refer to attributes seen in decision trees generated in this study. The labels *barely* and *very weakly* are explained in the text.

T. Menzies and J.S. Di Stefano are with the Lane Department of Computer Science, West Virginia University, PO Box 6109, Morgantown, WV, 26506-6109, USA emails: tim@menzies.com justin@lostportal.net.
Wp ref: 02/toolsai/sereuseletter

This research was conducted at West Virginia University under NASA contract NCC2-0979. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA IV&V Facility. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

II. Data Miners

Having described where our studies agreed, we now describe where the application of three data miners caused us to disagree with the conclusions of Morisio et.al. To do that, we must first describe our data miners. The goal of data mining is to find important patterns in data sets. Analyzing these data sets by hand is problematic at best, and can take substantial time and effort. It is both quicker and easier if a computer can be "taught" to search for these patterns.

In the 21^{st} century, data mining is a very mature field. Many powerful mining tools are freely available via the world wide web. This study applied three such mining tools to the Morisio et.al. data: the APRIORI *association rule learner* [2]; the J4.8 *decision tree learner* [3]; and the TAR2 *treatment learner* [4]. Our implementations of APRIORI and J4.8 come from the WEKA toolkit [3][1] while TAR2 came from the treatment learning download page[2]. The essential details of these tools are summarized below.

Decision tree learners find mappings between classes and non-class attributes. The class attributes in the Morisio et.al. data set were *successful reuse* and *failed reuse* while the non-class attributes are shown in the appendix. Figure 1 shows one example of such a mapping between class and non-class attributes. Note that of the nearly two dozen non-class attributes collected by Morisio et.al., only two appear in the decision tree. Decision tree learners seek the *most informative attribute ranges* that *splits* the training data into subsets with similar classes. The process repeats recursively for each subset and returns one sub-tree for each recursive call. Different decision tree learners use different criteria for splitting the training sets. The CART algorithm [5] used by Morisio et.al. uses the GINA index. In our study, we used J4.8 [3] which is the JAVA variant of C4.5 [6] that comes with the WEKA. C4.5 uses a splitting criteria based on information theory.

Decision tree learning can also be used to determine which attributes are most important using an *attribute removal experiment*. Decision trees have a root node which mentions the attribute range most useful in splitting the training data. If that attribute is removed from the training set and the learner is run again, then the root node seen in the new tree contains the *next most important* attribute. The results of attribute removal experiments on the Morisio et.al. data is shown in Figure 4:

- We say that an attribute is *barely supportive* if it is removed in the next attribute removal experiment but the classification accuracy does not change. Figure 4 shows that *Reuse Processes Introduced* and *Top Management Commitment* are barely supportive attributes.
- We say that an attribute is *very weak* if it first appears as a non-root node of a decision tree that is learnt very late in an attribute removal experiment; i.e. only after many more supportive attributes have been removed. *Key Reuse Roles Introduced* and *Non-Reuse Processes Modified* are very weak attributes since they only appeared in J4.8's decision trees after experiment 4 of Figure 4.

Association rule learners find attributes that commonly occur together in a training set. In the association $LHS \Longrightarrow RHS$, no attribute can appear on both sides of the association; i.e. $LHS \cap RHS = \emptyset$. The rule $LHS \Longrightarrow RHS$ holds in the example set with *confidence* c if $c\%$ of the examples that contain LHS also contain RHS; i.e. $c = \frac{|LHS \cup RHS| * 100}{|LHS|}$. The rule $LHS \Longrightarrow RHS$ has *support* s in the example set if $s\%$ of the examples contain $LHS \cup RHS$; i.e. $s = \frac{|LHS \cup RHS| * 100}{|D|}$ where $|D|$ is the number of examples. Association rule learners return rules with high confidence (e.g. $c > 90\%$).

[1] `http://www.cs.waikato.ac.nz/~ml/weka/`

[2] `http://www.ece.ubc.ca/twiki/bin/view/Softeng/TreatmentLearner`

#	experiment	classification accuracy
0	all attributes	96%
1	without *Human Factors*	79%
2	without *Reuse Processes Introduced*	79%
3	without *Production Type*	67%
4	without *Top management commitment*	67%

Fig. 4. Attribute removal experiments with J4.8. Each experiment I reruns the decision tree learning *without* the attribute found in the root of the tree seen in experiment $I - 1$.

The search for associations is often culled via first rejecting associations with low support. Association rule learners can be viewed as generalizations of decision tree learning since the latter restrict the RHS of rules to just one special class attribute while the former can add any number of attributes to the RHS. Example association rule learners include the implementation of the APRIORI [2] algorithm, available in the WEKA. Figure 5 shows the associations seen in the Morisio et.al. data.

1) Production Type=product-family $\Longrightarrow$ Rewards Policy=no
2) Production Type=product-family $\wedge$ Independent Team=no $\Longrightarrow$ Rewards Policy=no
3) Production Type=product-family $\wedge$ Top Management Commitment=yes $\Longrightarrow$ Rewards Policy=no
4) Software and Product=product $\Longrightarrow$ Rewards Policy=no
5) Production Type=product-family $\wedge$ Independent Team=no $\wedge$ When Assets Developed=justintime $\Longrightarrow$ Rewards Policy=no
6) Production Type=product-family $\wedge$ When Assets Developed=justintime $\Longrightarrow$ Rewards Policy=no $\wedge$ Independent Team=no
7) Independent Team=no $\wedge$ When Assets Developed=justintime $\Longrightarrow$ Rewards Policy=no $\wedge$ Production Type=product-family
8) When Assets Developed=justintime $\Longrightarrow$ Rewards Policy=no $\wedge$ Production Type=product-family $\wedge$ Independent Team=no
9) Top Management Commitment=yes $\wedge$ Rewards Policy=no $\Longrightarrow$ Production Type=product-family
10) When Assets Developed=justintime $\wedge$ Rewards Policy=no $\Longrightarrow$ Production Type=product-family $\wedge$ Independent Team=no

Fig. 5. Associations learnt by the WEKA's APRIORI implementation from the Morisio et.al.'s data.

Treatment learning seeks a *treatment* R_X that returns a subset of the training set $D' \subseteq D$ with *more* preferred classes and *less* undesired classes than in D. Here, D' contains all examples that don't contradict the treatment; i.e. $D' = \{d \in D : d \wedge R_X \not\vdash \bot\}$. The intuition here is that the treatment is some action that could improve the current situation. The TAR2 treatment learner requires the user to assign a numeric score to each class that represents how much a user likes/hates that class. For example, in the Morisio et.al. study, a *successful reuse project* would be worth more than an *unsuccessful* project. Treatment learning is different from decision tree learning in that treatment learners find treatments that *change* the class distribution while decision tree learners *describe* the different classes. The class descriptors found by decision tree learners are useful when studying the detailed features of a class. Treatments are useful when seeking actions that nudge the system towards preferred behavior. When TAR2 was applied to the Morisio et.al. data, it found that the following attribute ranges most selected for successful reuse projects:

- $Size\ of\ Baseline = L$; i.e. 100-500 KLOC;
- $Domain\ Analysis = yes$; i.e. domain analysis was performed;
- $Reuse\ Approach = tight$; i.e. reusable products are tightly coupled.

III. Results

Figure 2 shows where the conclusions offered by these learners differed from Morisio et.al. For example, Morisio et.al. commented

that they had to remove two attributes from their analysis "due to the low number of cases" [1, p355]. Accordingly, they removed $Application\ domain$ and $Size\ of\ baseline$. We found no reason to do the same: our learners functioned adequately when given all attributes.

Our learners found nothing interesting about $application\ domain$. However, contrary to the assumptions of Morisio et.al., *Size of baseline* was found to be a *very powerful* attribute for selecting for successful reuse:

- 100% of the 8 projects where $Size\ of\ baseline$ was "large"were judged to be reuse successes.

This result is simple to explain: reuse works best when most of the problem has already been mapped out and analysts just need to add in relatively small pieces, here and there. Support for our explanation comes from Abs et.al. [7, p21] who argue that a learning curve must be traversed before a module can be adapted. By the time you know enough to change a little of that module, you may as well have re-written 60% of it from scratch; see Figure 6. Note that when $Size\ of\ baseline$ is large, then the % changed by any new application is likely to be a small modification to the overall system. Hence, those changes would be cheap and easy to perform since they would fall into the lower left-hand side of Figure 6.

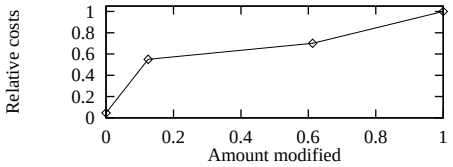

Fig. 6. COCOMO-II, the cost of reuse with X% changes. From [7].

The treatment learner also showed two other areas where our learners offer different conclusions to Morisio et.al.:

- 100% of the 11 projects where $Reuse\ approach$ were "tight" were judged to be reuse successes (Morisio et.al. never comment on the merits of *reuse approach*).
- 100% of the 9 projects that used $Domain\ analysis$ were judged to be reuse successes.

We also take issue with how Morisio et.al. learnt the decision tree of Figure 1. According to that decision tree:

- *Human Factors* is the best predictor for successful reuse projects.
- The only caveat to this pattern is the sub-tree testing for *Product Types* that are isolated systems (see the edge labelled "[1]" in Figure 1).

The merits of the *Product Type* sub-tree in Figure 1 is dubious. Without that sub-tree, the simple tree of Figure 7 accurately predicts for successful/unsuccessful reuse in $\frac{23}{24} \approx 96\%$ of the examples. With that sub-tree, the sub-tree catches the 4% special case where *Human Factors* does not predict for successful. While this seems a valid reason for adding the *Product Type* sub-tree, the empirical basis for it is very weak (one example). Conflating Figure 7 with the extra sub-tree of Figure 1 is not justified, in our view, based on this single example.

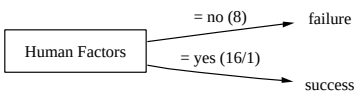

Fig. 7. A decision tree learnt from the Morisio et.al. data by the WEKA's J4.8 implementation. Numbers denote how many examples exercise some edge.

Morisio et.al. also argued that *Top Management Commitment* was a major factor in achieving a successful reuse program. While we find this claim to be intuitive, we are duty bound to report that none of learners found it to be predictive or associated with anything else. While *Top Management Commitment* appears in the learnt associations of Figure 5, none of those associations include successful or unsuccessful reuse. Further, recalling Figure 4, *Top Management Commitment* is a barely supportive attribute; i.e. the data of Morisio et.al. offers little evidence that this attribute is useful in predicting successful reuse.

IV. Discussion

There are several possible reasons why our analysis differs from theirs. Firstly, our mining tools only had access to the data published in Morisio et.al. and not the managers interviews in their analysis. That is, §6 of Morisio et.al. might be a summary of the discussions with project managers rather than conclusions drawn from their automatic analysis.

Secondly, our analysis of the data uses different machine learners to Morisio et.al's study. Some of these differences are minor: CART and C4.5/J4.8 come from the same family of learners and just differ on details of the splitting criteria. However, there are major differences in the other machine learners we used. For example, TAR2 is a recently invented learner by Menzies & Hu [4]. TAR2's report of *differences* between classes is a novel and succinct method of isolating the key factors that can most change a situation.

V. Methodological Implications for Software Research

The above analysis took less than two days and was enabled by the availability of free, fast, and mature data mining tools from the world wide web. Given the availability of these tools, we would recommend a change to the methodology of studies like (e.g.) Morisio et.al. In their approach no questions were asked after the data analysis period. Since these learners are so simple to use and readily available, we suggest a two-part interview process where the questions of part two are informed by the answers in part one.

Part one would be to "throw the net wide" and ask a large number of easy-to-answer questions. Interesting patterns could be then be found within the answers to part one using a range of machine learners.

The part two questions would be to "narrow the net" and focus on complex issues in controlled situations with a smaller group of users (perhaps users with more experience in the domain being studied). The part two questions should be designed to confirm or refute the patterns automatically detected by data miners after part one.

The advantage of this method is that unusual features can be found quickly (using data mining), then explored with the user group in the part two questionnaire. A variant of this approach (using pivot tables within a spreadsheet and not data miners) has recently been used with great effect at NASA [8], [9], [10].

References

[1] M. Morisio, M. Ezran, and C. Tully, "Success and failure factors in software reuse," *IEEE Transactions on Software Engineering*, vol. 28, no. 4, pp. 340–357, 2002.

[2] R.Agrawal, T.Imeilinski, and A.Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD Conference, Washington DC, USA*, 1993, Available from `http://citeseer.nj.nec.com/agrawal93mining.html`.

[3] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 1999.

Attribute	Value	#	Notes
Project id	1..∞		
Top management commitment	yes	20	top management reuse committed
	no	4	
Key reuse roles introduced	yes	19	>= 1 reuse role
	no	4	no reuse roles were introduced
Reuse process introduced	yes	15	>= 1 reuse process was
	no	8	no reuse process was introduced
Nonreuse process modified	yes	16	>= 1 nonreuse processes modified
	no	7	no nonreuse processes modified
Human factors	yes	16	human factors handled; e.g. via awareness, training, and motivation programs
	no	8	
Repository	yes	23	assets in repository tool
	no	0	

Fig. 8. High-level control variables - key high-level management decisions about a reuse program. Note that all 23 projects seen in this data set used a repository; i.e. this data set could never be used to refute claims that a repository is useless. Nevertheless, like Morisio et.al., we believe that reuse products have to be kept in some sort of repository to enable reuse.

[4] T. Menzies, E. Chiang, M. Feather, Y. Hu, and J.D. Kiper, "Condensing uncertainty via incremental treatment learning," in *Annals of Software Engineering*, 2002, Available from `http://tim.menzies.com/pdf/02itar2.pdf`.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and regression trees," Tech. Rep., Wadsworth International, Monterey, CA, 1984.

[6] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman, 1992, ISBN: 1558602380.

[7] C. Abts, B. Clark, S. Devnani-Chulani, E. Horowitz, R. Madachy, D. Reifer, R. Selby, and B. Steece, "COCOMO II model definition manual," Tech. Rep., Center for Software Engineering, USC,, 1998, `http://sunset.usc.edu/COCOMOII/cocomox.html#downloads`.

[8] R. Lutz and Carmen Mikulski, "Operational anomalies as a cause of safety-critical requirements evolution," *Journal of Systems and Software (to appear)*, 2003, Available from `http://www.cs.iastate.edu/~rlutz/publications/JSS02.ps`.

[9] R. Lutz and Carmen Mikulski, "Empirical analysis of safety-critical anomalies during operations," 2003, (submitted).

[10] Tim Menzies, Robyn Lutz, and Carmen Mikulski, "Better analysis of defect data at NASA," in *27th NASA SEL workshop on Software Engineering (submitted)*, 2002.

APPENDIX

Figures 8, 9 and 10 describe the attributes collected by the Morisio et.al. study.

Attribute	Value	#	Notes
Project id	1..∞		
Software staff	L	6	> 201 people on the project.
	M	9	51 . . . 200 people on the project.
	S	9	1 . . . 50 people on the project.
Overall Staff	X	10	> 501 people.
	L	7	201 . . . 500 people.
	M	5	51 . . . 200 people.
	S	2	1 . . . 50 people.
Production Type	product-family	20	projects related; evolve over time
	isolated	4	projects have little in common
Software and product	product	17	software is embedded in a product
	alone	4	software is standalone product
	process	2	software embedded in a process
SP maturity	high	6	CMM level 3 or higher
	medium	13	ISO 9001 certification or CMM level 2
	low	5	not high or medium
Application domain	TLC	7	telecommunications
	FMS	4	flight management systems
	ATC	1	air traffic control
	TS	1	train simulation
	TTC	7	train traffic control
	Bank	1	bank
	Book-keeping	1	book-keeping
	Measurement	1	management, control of measurements
	Space	1	aerospace applications
	Manufacturing	3	manufacturing
	SE-Tools	2	software tools
Type of software	Embedded-RT	6	embedded, real-time
	Non-Embedded-RT	2	non-embedded, real-time
	Technical	12	non-embedded, non-real-time, small DBMS, important control part
	Business	4	non-embedded, non-real-time, important DBMS, limited control part
Size of baseline	L	8	100 . . . 500 KLOC, > 100 person months
	M	13	10 . . . 100 KLOC, 10 . . . 100 person months
	S	2	< 10 KLOC, < 10 person months
Development Approach	OO	15	object oriented
	proc	8	procedural
Staff experience	high	7	> 5 years average
	medium	15	2 . . . 4 years average
	low	1	<= 1 year average

Fig. 9. State Variables - attributes over which a company has no control.

Attribute	Value	#	Notes
Project id	1..∞		
Reuse approach	loose	12	assets loosely coupled
	tight	11	assets coupled, used in groups)
Domain Analysis	yes	9	domain analysis was performed
	no	14	
Origin	ex-novo	4	assets are developed from scratch
	reeng	15	assets via reengineering old work
	as-is	4	old products used without change
Independent team	yes	2	independent team makes assets
	no	21	development projects makes assets
When assets built	before	7	well before they are reused
	just-in-time	16	just before they are reused
Qualification	yes	14	assets undergo a qualification process
	no	9	no defined qualification process
Configuration management	yes	16	configuration management used
	no	7	
Rewards policy	yes	3	a rewards policy for reuse in place
	no	21	no rewards policy in place
# of assets	1 to 20	5	number of assets in the repository
	21 to 50	3	
	51 to 100	8	
	100+	7	
Work-products	C		code
	D		design
	R		requirements

Fig. 10. Low-Level Control Variables - Specific approaches to the implementation of reuse. Numbers of work products are counted differently to the other parameters: i.e. C=10, D+C=4; R+D+C=9.