APPLICATION OF GENETIC PROGRAMMING IN SOFTWARE ENGINEERING EMPIRICAL DATA MODELLING

Athanasios Tsakonas and Georgios Dounias
Department of Financial and Management Engineering, University of the Aegean, 31 Fostini St., Chios, Greece

Keywords: Genetic programming, software engineering, data mining, effort estimation, defect prediction.

Abstract: Research in software engineering data analysis has only recently incorporated computational intelligence methodologies. Among these approaches, genetic programming retains a remarkable position, facilitating symbolic regression tasks. In this paper, we demonstrate the effectiveness of the genetic programming paradigm, in two major software engineering duties, effort estimation and defect prediction. We examine data domains from both the commercial and the scientific sector, for each task. The proposed model is proved superior to past literature works.

1 INTRODUCTION

Among the software engineering management duties, the effort estimation and the assessment of the software quality are still considered challenging tasks. Undoubtedly, an enhancement in software effort estimation could affect dramatically the overall project cost. Accordingly, appropriate quality assessment can also result into proper resource allocation and avoidance of further expenditures. In the past, various parametric models, such as COCOMO (Boehm, 1980), have been used for software effort estimation. To address software quality, the development of code metrics has taken place, aiming to facilitate the diagnosis of error-prone code (McCabe, 1976). Most of these models make use of past information, in an attempt to develop simple or complex mathematical expressions, usually in *ad-hoc* or exhaustive manners. On the other hand, the genetic programming paradigm (Koza, 1992), which has been around during the last decades, is applied in numerous domains, successfully addressing data mining and knowledge extraction tasks. The evolutionary search and the expression ability of the genetic programming have made this approach a popular research tool in symbolic regression problems. Consequently, this paradigm is expected to provide competitive results into the effort estimation and the prediction of error-prone code in software projects. In this paper, the genetic programming approach is applied into the two aforementioned software engineering tasks, effort estimation and defect prediction. Regarding effort estimation, we aim to provide symbolic regression, deriving accurate and small-sized expressions. For this, we examine datasets from both a commercial software domain, and a scientific one. In respect to prediction of error-prone code, the target is to produce discriminators between faulty and non-faulty modules, in the form of *if-then* rules. These formulas should be relatively small and accurate. Here too, two domains are examined, one from the commercial sector, and a scientific one. The genetic programming system we used, adopts recent developments in respect of the operation rates and the use of the validation set. The paper is organized as follows. Next section describes the scientific background, presenting in short the characteristics of the examined software engineering tasks and the genetic programming principle. Section 3 describes the data domains and the GP system design. The results and a followed discussion are shown in Section 4. The paper ends with our conclusions and future directions in Section 5.

2 BACKGROUND

One of the principal factors affecting the software project cost is the labor cost. Effort estimation and defect prediction aim to effectively allocate personnel in tasks and assure the software quality, both helping at reducing the software project cost.

Tsakonas A. and Dounias G. (2008).
APPLICATION OF GENETIC PROGRAMMING IN SOFTWARE ENGINEERING EMPIRICAL DATA MODELLING.
In *Proceedings of the Third International Conference on Software and Data Technologies - PL/DPS/KE*, pages 295-300
DOI: 10.5220/0001871302950300

The software project effort estimation involves the definition of resources, the management of the development activities and the calculation of the estimation risk (Lum et al., 2003). There are four types of effort estimation: *historical analogy, experts' decision, use of models and rules-of-thumb*. *Historical analogy* is used when there are similar data available from the past and it involves comparison, using measures or data that has been observed in previous software works. *Experts' decision* describes the estimates produced by a human expert based on what he has experienced from past projects. *Use of models* is estimating by mathematical or parametric cost models (which are derived usually statistically). Finally, *rules-of-thumb* are used for effort estimation, usually in the form of very simple mathematical equations. A data-mining task is therefore a historical analogy approach. Depending on the problem encountered, various data can be recorded for use. Two such domains are described in Table 1 and Table 2. On the other hand, aiming to enhance software quality and the resources allocation, various approaches of software code assessment have been developed, such as the static code metrics. These mostly come from the research of (McCabe, 1976) and (Halstead, 1977). McCabe's metrics are the *cyclomatic complexity*, the *essential complexity*, the *design complexity* and the *number of lines of code*. Cyclomatic complexity, *v(G)*, counts the number of linearly independent paths:

$$v(G) = e - n + 2 \quad (1)$$

where *G* is the module's flow-graph, *e* are the arcs in the flow-graph, and *n* are the nodes. The essential complexity, *ev(G)*, measures the potential shrink of the flow-graph:

$$ev(G) = v(G) - m \quad (2)$$

where *m* are the sub-flow-graphs of G that are *D-structured primes* (Fenton and Pfleeger 1977). Design complexity, *iv(G)*, is the cyclomatic complexity of a module's reduced flow-graph (McCabe, 1976). *Lines of code* are counted based on to McCabe's conventions. Halstead's measures are classified into three groups: *base* measures, *derived* measures and measures *related to lines of code*. Table 4 summarizes the aforementioned measures. Other frameworks are often adopted for defect prediction, regarding special needs of a project, such as the Quality Improvement Paradigm (QIP) (Basili et al., 1994), in order to apply a systematic quality assurance, by reusing the experience. This framework involves software measurement and analysis from carefully designed experiments. The metrics selected from such an approach, are shown in Table 3. Genetic programming - GP (Koza, 1992) is an advance to the genetic algorithms paradigm. This paradigm enables symbolic regression, i.e. the search for complex solutions in the form of mathematical formulas. Further research extended this concept to calculate any kind of boolean or programming expression. Although widely used during the last two decades, the GP has only recently been applied into software engineering tasks, such as the evaluation of software project managers (Boetticher et al., 2006). In order to improve the search and expression ability, the function set that composes a GP desired solution is often traced with the aid of a grammar. Here, we apply grammar-guided search for the two defect prediction tasks (e.g. classification problems), using a context-free grammar for the production of *if-then* hierarchical rules (Tsakonas and Dounias, 2002).

3 SYSTEM DESIGN

3.1 Domain Description

The following detailed description of the datasets aims to denote the high variance in their composition (in terms of the features used), which could assist in generalising on the performance of our system. Regarding effort estimation, we first address the *Desharnais* domain. This is consisted of Canadian software house data for commercial projects (Desharnais, 1989). It contains 81 records. Table 1 summarizes the variables' descriptions.

Table 1: Desharnais data feature description.

Variable	Explanation
TEM	Team experience (in years)
MNG	Manager experience (in years)
YER	Year end
LEN	Length
TRN	System's basic transactions
ENT	# entities in the system's data model
PTA	Adjusted points
ENV	Envergure
PTN	Non-adjusted points
LAN	Language
Person-months	Effort in months (regression target)

The *NASA93* dataset consists of 93 NASA projects from different centres for projects taken place between 1971-1987. This software comes from the aerospace domain. There are 17 attributes that are all numeric: 15 attributes are the effort multipliers, one

is the Lines-of-Code (LOC) and one attribute is the actual development effort. The LOC variable has been estimated directly or computed beforehand, using function point analysis (Dreger 1989).

Table 2: NASA93 data description.

Variable	Explanation
rely	Required software reliability
data	Data base size
cplx	Process complexity
time	Time constraint for CPU
stor	Main memory constraint
virt	Machine volatility
turn	Turnaround time
acap	Analysts capability
aexp	Application experience
pcap	Programmers capability
vexp	Virtual machine experience
lexp	Language experience
modp	Modern programming practices
tool	Use of software tools
sced	Schedule constraint
ln(KSLOC)	Software size lines-of-code
ln(months)	Effort in months (regression target)

Table 3: Datatrieve data description – measurements according to QIP.

Variable	Explanation
LOC60	Number of LOC of module m in version 6.0
LOC61	Number of LOC of module m in version 6.1.
aLOC	LOC added to module m in v.6.1
dLOC	LOC deleted from module m in v.6.0
DBLK	Number of different blocks module m between versions 6.0 and 6.1
MRAT	Rate of modification of module m, equals to $(aLOC + dLOC)/(LOC60 + aLOC)$
MKNW	Team's knowledge on module m
rLOC	LOC of module m in v.6.0 reused in v.6.1
Class	0 for non-faulty modules, 1 for the rest

The task is to tune a new cost model, for a given background knowledge. Table 2 describes the dataset features. The *Datatrieve* data set follows the QIP approach and it contains 130 records (Morasca and Ruhe, 2000). This data is from the Datatrieve product, which was undergoing both adaptive and corrective maintenance. The objective of the data analysis is to examine whether it was possible to predict the faulty modules based on a set of measures that have been collected on the project. Table 3 summarizes the descriptions of the calculated features. The JM1 domain is consisted of 10,885 records that contain measurements from NASA's C modules of a real time predictive ground system that uses simulations to generate predictions.

Table 4: NASA JM1 data description – McCabe and Halsted metrics.

Variable	Explanation
mu_1	number of unique operators
mu_2	number of unique operands
N_1	total occurrences of operators
N_2	total occurrences of operands
N	$N = N_1 + N_2$ (length)
mu	$mu = mu_1 + mu_2$ (vocabulary)
P	$P = V = N \log_2(mu)$ (volume)
V^*	$V^* = (2 + mu_2') \log_2(2 + mu_2')$
L	$L = V^* / N$ (program length)
D	$D = 1 / L$ (difficulty)
L'	$L' = 1 / D$
I	$I = L' V'$ (intelligence)
E	$E = V / L$ (effort to write program)
T	T = E / 18
lco	Halstead count of lines of code
lcm	Halstead count of lines of comments
lcb	Halstead count of blank lines
lcc	Halstead count of LOC and comments
brc	Branch count of the flow-graph
LOC	McCabe's lines of code
v(G)	$v(G) = e - n + 2$
ev(G)	$ev(G) = v(G) - m$
iv(G)	$iv(G) = v(G)$, module's reduced flow-graph
class	binary, whether the case had defects or not

This data has been publicly available by NASA's Metrics Data Program (MDP) (`http://mdp.ivv.nasa.gov/`). It has 22 input features and one binary output, whether the case had defects or not. Table 4 summarizes the feature descriptions. All these data are publicly available from the PROMISE reporitory (`http:// promise.site.uottawa.ca/SERepository/`).

3.2 System Setup

We normalized each data set linearly (i.e. using the *min-max* transformation) in the range [-1,1], in order to improve the search process. The genetic programming parameters are shown in Table 5. We applied 10-fold cross validation, having each time a 10% of the data outside of the training procedure, to be checked as test set. A validation set is used to avoid overfitting to the training set. In order to promote a solution that carries a higher value in the training set, the candidate should either also have a higher value in the validation set or the absolute difference between validation fitness and training

fitness score should be smaller, the latter being a property empirically driven during our experiments, found to increase generalization strength. For the regression problems (effort estimation), as fitness measure we have used the *mean magnitude relative error, MMRE,* (eq. 3)*:*

$$MMRE=\frac{100}{n}\sum_{i=0}^{n-1}\left|\frac{y_i-\hat{y}_i}{y_i}\right| \quad (3)$$

Due to critics on using a single measure for evaluating models (Foss et al., 2003), as well as for comparison reasons, the measures (4)-(6) are also calculated:

$$RRSE=\sqrt{\sum_{i=0}^{n-1}\left(y_i-\hat{y}_i\right)^2/\sum_{i=0}^{n-1}\left(y_i-\overline{y}_i\right)^2} \quad (4)$$

$$RAE=\sum_{i=0}^{n-1}\left|y_i-\hat{y}_i\right|/\sum_{i=0}^{n-1}\left|y_i-\overline{y}_i\right| \quad (5)$$

$$\mathrm{PRED}(r)=\frac{100}{n}\sum_{i=0}^{n-1}\begin{cases}1 \; if \; \left|\frac{y_i-\hat{y}_i}{y_i}\right|\le\frac{r}{100}\\ 0 \; otherwise\end{cases} \quad (6)$$

In all above equations, y_i : actual value of case *i*, $\hat{y}_i$: estimated value of case *i*, $\overline{y}_i$: mean value of test set cases, n : number of cases in test set, r **:** value (range) for which the *PRED* function is calculated, usually being equal to *25* or *30.* For the classification problems (defect prediction) we selected the fitness function as follows. When the system classifies a case, the following situations may happen (Koza, 1992): *tp:* true positive, *fp:* false positive, *tn:* true negative and, *fn:* false negative, according to. Taking in respect these outcomes, we used the following measure as fitness function (Berlanga et al. 2005):

$$\mathrm{support}=pd\cdot TNRate=\frac{tp}{tp+fn}\cdot\frac{tn}{tn+fp} \quad (7)$$

As other measures have been proposed in literature (Menzies et al. 2007), they are also calculated in our experiments; these are the *precision,* the *pf* (probability of failure) and the *accuracy*:

$$prec=\frac{tp}{fp+tp} \quad (8)$$

$$pf=\frac{fp}{tn+fp} \quad (9)$$

$$accuracy=\frac{tp+tn}{tp+tn+fp+fn} \quad (10)$$

Table 5: Genetic programming parameter setup.

Parameter	Value
Population	9,000 individuals
GP implementation	Steady-state GP (regression problems) Steady-state grammar-guided GP (classification problems)
Selection	Tournament of 6 with elitist strategy
Training	10-fold cross validation
Crossover/ Mutation/ Copy	Adaptive (Tsakonas and Dounias, 2007)
Maximum size	650 nodes
Maximum generations	200

4 RESULTS AND DISCUSSION

In order to allow direct comparison between the proposed system and the past work, we selected to report our results, for each dataset, using the measures that past literature has adopted. Addressing the effort estimation task, we first examine the *Desharnais* domain. In Table 6, the results from our 10-fold validation are presented together with test results from WEKA's (`http://www.cs.waikato.ac.nz/~ml/weka/`) linear regression system (10-fold validation) and the best model from the work in (Shepperd and Schofield, 1997). The latter, previous work results are the only ones found in literature, concerning this data set. As it can be seen, our system maintains the highest values in all compared accuracy measures. The following simple solution was derived in fold #1, it uses the adjusted points and the number of entities, and it achieved 100% *PRED(25)*.

$$_N(personhours)=\,_N(PTA)-0.1\cdot\,_N(ENT) \quad (11)$$

where $_N(\cdot)$ denotes that the normalized values of the corresponding variables are used.

Table 6: Results comparison for the Desharnais effort estimation domain.

	RRSE	RAE	MMRE	Pred(25)
WEKA L-R	67.7	64.99	n/a	n/a
Shepperd	n/a	n/a	64	42
This work	67.15	62.01	31.32	77.78

The next domain to examine in regard of effort estimation is the *NASA93* data set. This problem has only been addressed in (Menzies et al., 2006). Table 7 compares our 10-fold validation results to the best of the models reported in that work. As it can be

seen, our model has the highest Pred(30) and a competitive MMRE, whilst the standard deviation remains the smallest.

Table 7: Results comparison for the NASA93 effort estimation domain.

	Pred(30)	MMRE	
		mean %	sd%
nasa93:center.2	83	22	38
nasa93:project.Y	78	22	20
nasa93:fg.g	65	32	39
This work	85.78	23.25	17.31

The following small solution that was derived in fold #2, uses only one feature (apart *KSLOC*), and it achieved 100% *PRED(25)*.

$$_N(\ln(months)) = _N(\ln(KSLOC)) + 0.2 \cdot _N(cplx) \quad (12)$$

where $_N(\cdot)$ is the symbol for the normalized values of the corresponding variables, as previously.

Regarding the defect prediction task, we first address the *Datatrieve* problem. The only past research where this data is examined is in (Morasca and Ruhe, 2000), using logistic regression and rough sets (*leave-one-out* cross validation). We have included the same measures for comparison reasons. Table 8 presents analytic results of our model. Table 9 compares the three systems in terms of classification *accuracy*, *precision* and *recall*. As it can be seen, our system outperformed the previous two in terms of accuracy. Our GP system has shown more balanced results than the past models, since it achieves high precision rate and good recall rate whilst having the higher overall accuracy of all three systems.

Table 8: Results in Datatrieve problem.

GP	Predicted non-faulty	Predicted faulty	Total
Actual non-faulty	90.0%	3.8%	93.8%
Actual faulty	1.5%	4.6%	6.2%
Total	91.5%	8.5%	100%

One derived GP solution is shown in Figure 1, in prefix notation. This is the #1 fold best solution and it achieved 100% accuracy in the test set. As it can be seen, all the available features from Table 3 - except LOC60- are used for making the classification. In this formula, CLS0 corresponds to non-faulty class and CLS1 to the faulty one.

Table 9: Results comparison for the Datatrieve defect prediction domain.

	Precision	Recall (*pd*)	Accuracy
Logistic Regression	90.6%	21.3%	70.8%
Rough sets	50.0%	70.6%	93.8%
This work	75.0%	54.5%	94.6%

```
IF> MKNW 0.76 (IF< aLOC -0.45 (IF< LOC61 -0.35 CLS0 CLS1) CLS1) (IF= dLOC 1.05 (IF> LOC61 0.53 CLS0 (IF< CLS1 -0.12 CLS1 (IF= DBLK 1.06 CLS0 (IF= MRAT 0.29 CLS0 CLS0)))) CLS0))
```

Figure 1: Hierarchical rule tree for the Datatrieve domain.

The next defect prediction domain to address is the *NASA JM1* data set. This problem has been addressed previously only in (Menzies et al., 2004). Table 10 compares the 10-fold validation mean, with these literature results. As it can be seen, our system outperformed the previous model regarding *pd* (probability of detection).

Table 10: Results comparison for the NASA JM1 defect prediction domain.

	pd	*pf*	Precision	Accuracy
Menzies et al.	25	18	n/a	n/a
This work	64.2	32.01	32.53	67.26

The solution in Figure 2, presented here in prefix notation, derived during fold #10, and it has *pd* equal to 66.67%, *pf* equal to 32.2%, *precision* 33.1% and *accuracy* 67.58%. Here too in this formula, CLS0 corresponds to non-faulty class and CLS1 to the faulty one. For the rest variables in this expression, the reader is referred to Table 4.

5 CONCLUSIONS

This work presented the application of the genetic programming for software engineering data analysis. The problems addressed were selected in respect of the data availability and the potential of applying historical analogy estimation. The genetic programming system incorporated recent advances in respect of the operator rates and the use of the validation set. Two tasks were designated aiming to contribute in software cost control and software

quality. The first task was to produce effort estimators using GP symbolic regression. The second task was to produce classification rules using grammar-guided GP for if-then rules. For each problem, two available datasets were examined. In the regression task of effort estimation, the system produced short and comprehensible mathematical expressions that outperformed previous models. In the classification task of defect prediction, the system produced the best-so-far or otherwise competitive results, while succeeding in deriving small hierarchical rule trees. Although the system has been proved capable of producing highly accurate and meaningful results for each case, this success is only domain dependent, e.g. it seems that due to the different dataset compositions, there is no ability to drive overall conclusions on each of the two tasks (effort estimation, defect prediction). Further research involves the application of other computational intelligent approach in these domains, such as neuro-fuzzy rule-based systems, as well as the application of genetic programming into other related software engineering issues.

```
(IF= (% D 0.35) 0.01 CLS1 (IF> I -0.84
(IF> I -0.84 (IF> LOC -0.98 CLS1 CLS0)
(IF> LOC -0.98 CLS1 CLS0)) (IF> I -0.85
CLS1 (IF< LBL -0.98 (IF> L -0.53 CLS0
(IF> EVG -0.83 CLS1 (IF= (% D 0.35)
0.01 CLS1 (IF> I -0.84 (IF> I -0.84
(IF> LOC -0.98 CLS1 CLS0) (IF> LOC -
0.98 CLS1 CLS0)) (IF> I -0.85 CLS1 (IF<
IVG 0.06 (IF> I -0.80 (IF> I -0.83 CLS1
(IF= (% D 0.36) 0.01 CLS1 (IF> I -0.85
CLS1 (IF> LOC -0.98 CLS1 CLS0)))) (IF>
L -0.53 CLS0 (IF> I -0.87 CLS1 (IF> I -
0.85 CLS1 (IF> TON -0.82 CLS1 (IF> LOC
-0.98 CLS1 CLS0)))))) CLS1)))))
CLS1))))
```

Figure 2: Hierarchical rule tree for the NASA JM1 domain.

REFERENCES

Basili V.R., Caldiera G., and Rombach H.D., 1994, Experience Factory, in *Marciniak, J.J., Ed., Encyclopedia of Software Engineering*, Vol. 1, pp. 469-476, John Wiley & Sons.

Berlanga F.J., del Jesus M.J., Herrera F., 2005, Learning compact fuzzy rule-based classification systems with genetic programming, *in EUSFLAT05, 4th Conf. of the European Society for Fuzzy Logic and Technology*, Barcelona, pp. 1027-1032.

Boehm B., 1981, *Software Engineering Economics*, Prentice-Hall.

Boetticher, G., Lokhandwala, N., James C. Helm, 2006, Understanding the Human Estimator, *in 2nd Int'l Predictive Models in Soft. Eng.Workshop*, Philadelphia, PA, Sep. 2006.

Desharnais J.M., *Analyse statistique de la productivite des projets informatique a partie de la technique des point des fonction*, 1989, MSc thesis, Univ. of Montreal.

Dreger J., 1989, *Function Point Analysis*, Englewood Cliffs, NJ, Prentice Hall.

Fenton N.E., and Pfleeger S., 1997, *Software Metrics: A Rigorous and Practical Approach*, Thompson Press.

Foss T.,Stensrud E., Kitchenham B., Myrtveit I., 2003, A Simulation Study of the Model Evaluation Criterion MMRE, *IEEE Trans. on Soft. Eng.*, 29:11, Nov. 2003.

Halstead M., 1977, *Elements of Software Science*, Elsevier.

Koza J.R., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA, MIT Press.

Lum K., Bramble M., Hihn J., Hackney J., Khorrami M. and Monson E., 2003, *Handbook of Software Cost Estimation*, Jet Propulsion Laboratory, Pasadena, CA, USA.

McCabe T., 1976, A Complexity Measure, *IEEE Trans. Software Eng.*,2:4, pp. 308-320.

Menzies T., DiStefano J., Orrego A., and Chapman R., 2004, Assessing Predictors of Software Defects, *in Proc. PSM-2004, Workshop in Predictive Software Model*s, Chicago, IL.

Menzies T, Chen Z., Hihn J., and Lum K., 2006, Selecting Best Practices for Effort Estimation, *IEEE Trans. Software Eng*. 32:11, Nov 2006.

Menzies T., Dekhtyar A., Distefano J., Greenwald J., 2007, Problems with Precision: A Response to "Comments on; Data Mining Static Code Attributes to Learn Defect Predictors, *IEEE Trans. on Soft. Eng.*, 33: 9, Sept. 2007 pp. 637 - 640.

Morasca S., and Ruhe G., 2000, A hybrid approach to analyze empirical software engineering data and its application to predict module fault-proneness in maintenance, *J. of Systems and Software*, 53:3, Sep. 2000, pp. 225 – 237, Elsevier

Shepperd M. J., and Schofield C.,1997, Estimating software project effort using analogies, *IEEE Trans. on Soft. Eng.*, 23, pp. 736-743.

Tsakonas A., Dounias G., Hierarchical Classification Trees Using Type-Constrained Genetic Programming, *in Proc. of 1st Intl. IEEE Symposium in Intelligent Systems,* Varna, Bulgaria, 2002.

Tsakonas A., and Dounias G., Evolving Neural-Symbolic Systems Guided by Adaptive Training Schemes: Applications in Finance, *App. Art. Intell.*, 21:7, 2007, pp. 681-706.