SYMBOLIC REGRESSION WITH SAMPLING

Michael Kommenda[a], Gabriel K. Kronberger[b], Michael Affenzeller[c], Stephan M. Winkler[d], Christoph Feilmayr[e], Stefan Wagner[f]

[a – d, f] Upper Austria University of Applied Sciences
School for Informatics, Communications, and Media
Heuristic and Evolutionary Algorithms Laboratory
Softwarepark 11, 4232 Hagenberg, Austria

[e] voestalpine Stahl GmbH
Research & Development Ironmaking
4020 Linz, Austria

[a] michael.kommenda@fh-hagenberg.at, [b] gabriel.kronberger@fh-hagenberg.at, [c] michael.affenzeller@fh-hagenberg.at, [d] stephan.winkler@fh-hagenberg.at, [e] christoph.feilmayr@voestalpine.com, [f] stefan.wagner@fh-hagenberg.at,

ABSTRACT
In this paper a way of improving the performance of genetic programming (GP) for regression tasks is presented. In general, most of the execution time is consumed during the evaluation step of an individual. Hence reducing the number of samples which are evaluated during the learning phase of the algorithm significantly reduces its execution time. A reduction of the available training samples might hamper the algorithm in its capability to learn the desired correlation. For this reason our approach evaluates each solution only on a randomly chosen part of all training samples, which is selected before the evaluation step. In the result section runs with different parameter settings of our approach and traditional genetic programming algorithms are compared regarding the solution quality and execution time to each other.

Keywords: Genetic Programming, Symbolic Regression, Sampling, Machine Learning, Performance Analysis

1. INTRODUCTION

1.1. Regression

Regression analysis is the task of modeling a relationship between a dependent (target) variable y and several independent (input) variables *x* in a dataset. Thus we want to get function f which calculates the target variable *y* using the input variables *x* and different weights *w* (1). The identified model is all the better the smaller the error term ε.

$$y = f(x, w) + \varepsilon \quad (1)$$

Regression analysis is often performed using supervised machine learning algorithms such as support vector machines (SVMs), artificial neuronal networks (ANNs) and genetic programming (GP) or statistical methods such as linear and polynomial regression. All of these methods have in common that available data is divided into a training and test partition. The training partition is used to learn the model and afterwards the generalization capabilities of the selected models are evaluated on the test partition, which must not have been used during the training. Some algorithms additionally take a part of the training partition for parameter optimization or model selection. This part of the training partition is referred to as validation partition.

After the identification of a model its performance must be measured. This is mostly done using the mean squared error (MSE) between the predicted and the original values of the target variable. Other correlation measures like the Pearson correlation coefficient (R^2), Spearman's rank correlation coefficient or variations of the MSE are also commonly used.

1.2. Symbolic Regression by Genetic Programming

Genetic programming (GP), an extension of genetic algorithms, was first studied at length by John Koza (1992). In contrast to the goal of genetic algorithms, finding a fixed length vector of predefined symbols, GP tries to find a variable length program to solve a given problem. The identified program is often represented as structure tree of a computer program, similar to symbolic-expressions of functional programming languages. Since GP evolves variable length programs no assumption about the structure of the programs needs to be made. GP is regarded as an evolutionary and population based optimization technique and the algorithmic steps are described in the following.

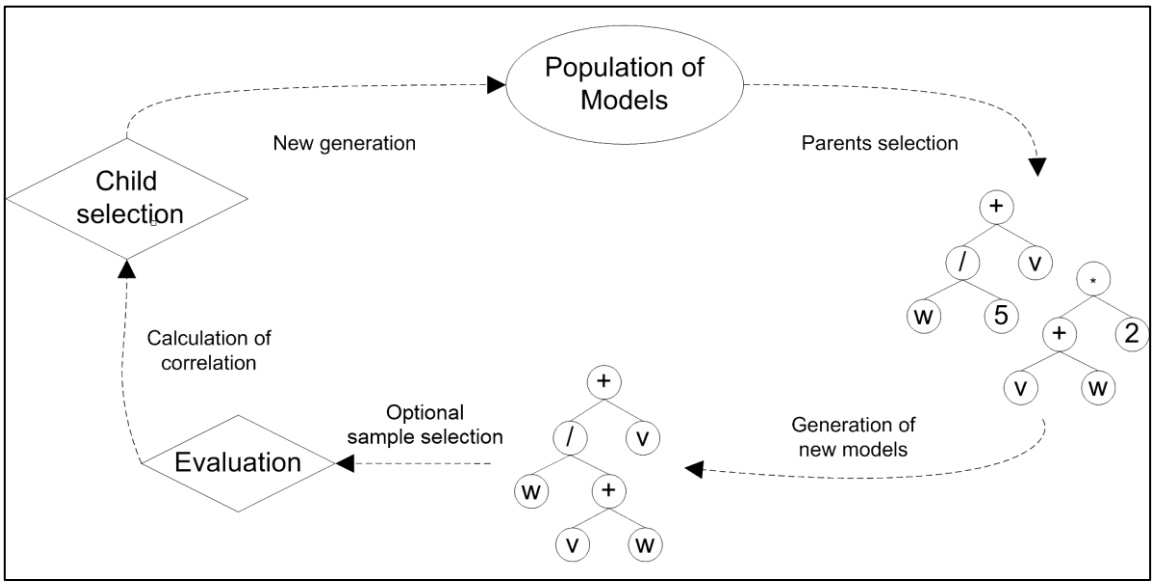

Figure 1: Schematic Representation of the GP algorithm

The population is first initialized with random individuals (structure trees), whose quality is calculated by a problem dependent fitness function. In the case of symbolic regression the MSE is mostly used as fitness function. Every generation parts of the population are replaced by new child individuals, created by combining the information of two parent individuals, e.g., merging parts of the parent structure trees. Afterwards the newly created child individual is mutated with a given probability to induce additional diversity in the population. The probability for an individual to be selected as parent is usually correlated to its fitness. The algorithm finishes if a given termination criterion is meet, e.g. a maximum number of generations has been evaluated. A schematic representation of a GP algorithm is shown in Figure 1.

When using GP for regression tasks the individuals are represented as structure trees, where each non terminal node represents a mathematical function and each terminal node represents either an input variable or a constant. Therefore the whole tree represents a mathematical formula as described in formula 1. During the evaluation step of the algorithm the estimated values of the target variable must be calculated for each created model. As this step is very time consuming, especially for large datasets, reducing the number of evaluated samples without losing predictive power is advantageous.

2. ALGORITHM EXTENSIONS

We introduced the relative number of evaluated samples as an additional parameter in the GP algorithm. This parameter ranges from [0, 1] and states the relative number of samples of the training partition that should be used in the algorithm. For example if the relative number of evaluated samples is 0.2 only 1/5 of the training partition is considered during the evaluation of the model. But overall the algorithm considers the whole information present in the trainings data, because this 1/5 is chosen repeatedly.

There are two possibilities how this reduction of the training partition could be implemented. In the first algorithmic extension all individuals are evaluated on the same part of the training in each generation. The selected training samples are changed only during the generational step of the genetic algorithm. In contrast to this, the other possibility is to evaluate all individuals on a different part of the training set by randomly selecting the training samples before each evaluation.

If every individual should be evaluated on the same training samples, the training partition is shuffled in every generation. The shuffling was implemented using a Fisher-Yates algorithm as described by Richard Durstenfeld (1964). Every individual is afterwards evaluated on the first *K* samples of the shuffled dataset. *K* defines the number of samples that should be used for evaluation and is calculated as the total number of training samples *N* multiplied with the relative number of evaluated samples.

The other possibility is that every individual is evaluated on a different, randomly chosen, part of the training samples. In this case it would be inefficient to shuffle the whole training data before each evaluation. Therefore we select a sequence of K unique samples between the training samples start and end. This problem is equivalent to the problem of picking *K* items from a collection of *N* items and can be efficiently solved using the selection sampling technique described by Knuth (1997). The selection sampling technique works by iterating over all samples until *K* samples are selected and is shown in Table 1; *N* defines the total number of samples, *n* the number of remaining samples, *K* the number of samples to select and *k* the number of already selected samples.

Table 1: Pseudo Code of the Selection Sampling Technique

```
While k < K
    Select actual sample with probability (K-k) / n
    If sample is selected
        Increase k
    Decrease n
    Step to next sample
End
```

It can be shown that this algorithm produces an unbiased random subset of the total samples by varying the probability to select a sample. The probability is equal to the relative number of evaluated samples for the first sample. It increases while the number of remaining samples *n* decreases and decreases if a sample got selected. All samples are selected with the same probability and exactly *K* samples are selected.

These two approaches to reduce the number of evaluated samples are only used for the training partition and so for the learning phase of the GP. In contrast all generated models are always evaluated on the whole validation partition, because the best performing model on the validation partition is returned as the result of the algorithm. Therefore the comparison value (MSE on the validation partition) must not be falsified, which forbids virtually reducing the size of the validation partition. Reducing the size of the test partition is also not reasonable, because it would change the estimation of the generalization capabilities of a model.

The reasons why reducing the number of evaluations is desirable are listed in Poli and McPhee (2008). In the first place the execution time of the GP algorithm is drastically reduced and in addition it is less likely that a specialized individual dominates the whole population. We could verify these advantages by achieving a significant speed up in terms of execution time and no drawback in terms of the quality of the identified solutions.

3. EXPERIMENTAL SETUP

The tests for these algorithm adaptations were performed on a regression dataset from Dow Chemical. The dataset was used in the symbolic regression competition, a side event of the EvoStar 2010 conference, and is publicly available at http://casnew.iti.upv.es/index.php/evocompetitions/105-symregcompetition. It contains 57 different input variables of a chemical real world process at Dow Chemical and 1066 samples. The sizes of the different partitions were 375 samples for training, 375 for validation and 316 for testing.

Furthermore we prepared a second dataset that contains data collected from an iron ore reduction process of our project partner voestalpine. It contains 5449 samples and 23 input variables describing the input material, products and the state of the blast furnace. A detailed description of the blast furnace process can be found in Kronberger et al. (2009). In this larger dataset 1900 samples were used for training, as well as 1900 for validation and 1800 samples for the test partition.

3.1. Experiments

We tested the described improvements on the Dow Chemical and on the voestalpine dataset. The major difference between the different algorithm runs was the population size parameter, which was chosen 1000 or 5000 respectively. The population size directly affects the execution time of the run because the number of evaluations during the algorithm run depends on the population size and on the number of generations. In addition the relative number of evaluated samples was varied between 0.1, 0.5, and 1.0. The parameter settings of the GP algorithm are summarized in Table 2.

These two changing parameters led to six different parameter combinations which were tested with the two algorithmic adaptations; whether all samples are evaluated on a differently chosen part of the training samples or if the part of the training samples is fixed during each generation.

Table 2: GP Algorithm Parameters

Population size	1000, 5000
Generations	300
Relative number of evaluated samples	0.1, 0.5 ,1
Mutation rate	0.15
Max tree height	10
Max tree size	100
Elites	1
Crossover	SubTreeCrossover
Selection	Tournament selection
Tournament size	5

3.2. Implementation

The approaches described in this paper have been implemented using the most recent version (3.3) of HeuristicLab (Wagner, 2009). HeuristicLab is a generic framework for modeling, executing and comparing different heuristic optimization techniques and provides plenty of functions for result analysis and evaluation. Another advantage is that all operators necessary for using GP for symbolic regression are already available. A binary version of HeuristicLab is available at http://dev.heuristiclab.com/trac/hl/core.

4. RESULTS

In this section the results regarding the execution time of the algorithm and the solution quality (MSE on the test partition of the best model per algorithm) on the Dow Chemical and the voestalpine dataset are shown. The main parameters were the sample selection strategy (every generation or every evaluation), the relative number of evaluated samples, and the population size. As the GP process is stochastic we repeated each parameter setting 40 times.

4.1. Execution time

Figure 2 shows a boxplot of the different execution times of the algorithm with a population size of 1000. The x-axis indicates the sample selection strategy, whether the training samples were the same for one generation or if the training samples varied for every evaluation, and the relative number of evaluated samples parameter value. The y-axis shows the execution time of all 40 repetitions as boxes. The results regarding the execution time for GP algorithms with a population size of 5000 are shown in Figure 3. It is noticeable that the execution time of runs that use the same 10% of the training partition for all models in a generation (Figure 3, first box), spreads strongly, but the median (dotted line in the boxplot) is as expected lower than the median of the runs which use more samples of the training partition. The results on the voestalpine dataset are shown in Figure 4 and 5. The runs illustrate the same correlation between the relative number of evaluated samples and the execution time, except that the average execution time is longer because of the larger training, validation, and test partitions.

Whenever only parts of training set are used, the runtime drops significantly. The validation and test partition have the same amount of samples for every algorithmic setting. Additionally, every structure tree must be interpreted before it can be evaluated. The interpretation is only dependent on the tree size and so it is not affected by the reduction of the evaluated samples. These two factors specify a lower bound on the execution time.

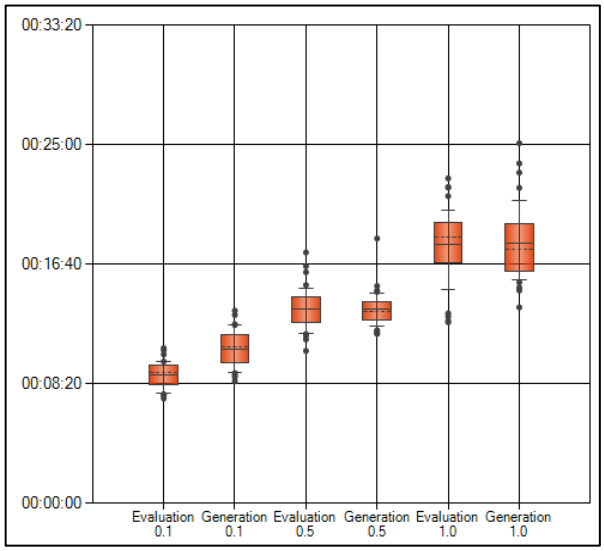

Figure 2: Boxplot of the Execution Times of GP Runs on the Dow Chemical Dataset with Population Size 1000

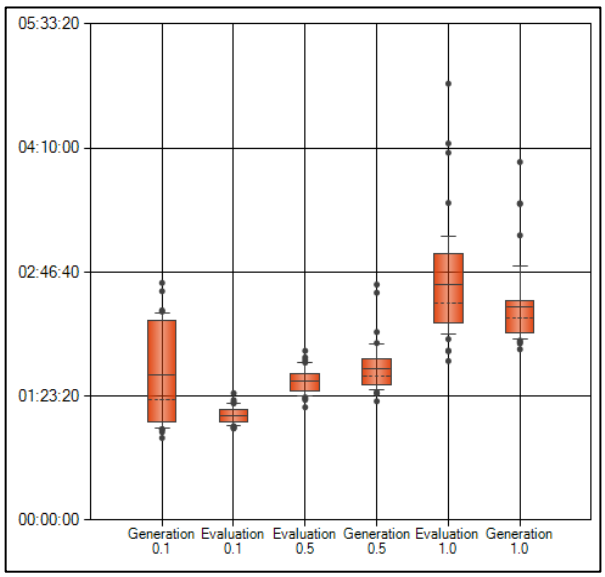

Figure 3: Boxplot of the Execution Times of GP Runs on the Dow Chemical Dataset with Population Size 5000

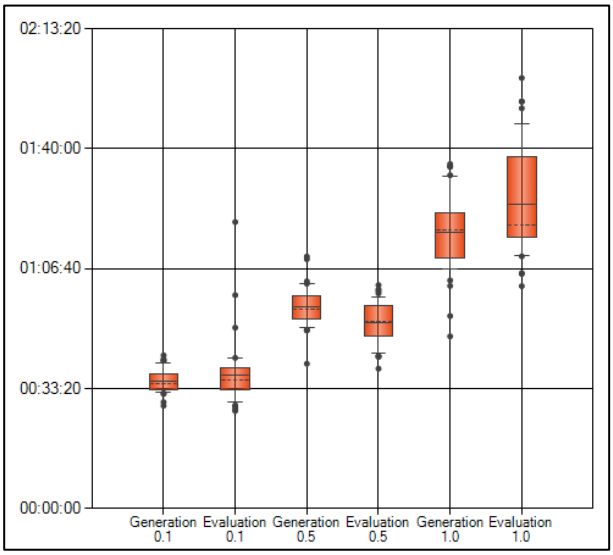

Figure 4: Boxplot of the Execution Times of GP Runs on the voestalpine Dataset with Population Size 1000

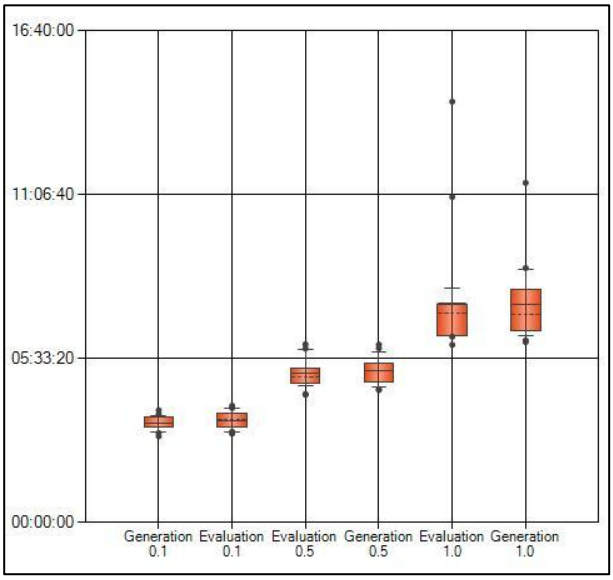

Figure 5: Boxplot of the Execution Times of GP Runs on the voestalpine Dataset with Population Size 5000

As it is shown in the figures above the runtime is drastically affected by the number of evaluated samples in each generation. However, the reduction would be useless, if the achieved quality was not competitive to the quality achieved by evaluating all available training samples.

4.2. Generalization error

The generalization error of the identified models is estimated by evaluating the models on the test partition. The validation partition could not be used for this task because the identified model is selected as the best performing model on the validation partition. The median MSE and its standard deviation of all 40 algorithm runs, per sample selection strategy and population size on the Dow Chemical dataset, are listed in Table 2. There is no obvious difference when only parts of the training partition are used to learn the models. Additionally the standard deviation of the MSE is bigger when more samples are used for training (relative number of evaluated samples 1.0). The reason for this is that it is more likely that the models are overfitted when using more training samples.

Table 2: Median and the standard deviation of the MSE on the test partition of the Dow Chemical dataset

Sample selection strategy	Population size	Median (MSE)	StDev (MSE)
Generation 0.1	1000	0.0578	0.0080
	5000	0.0520	0.0078
Generation 0.5	1000	0.0637	0.0091
	5000	0.0571	0.0329
Generation 1.0	1000	0.0609	0.2512
	5000	0.0598	0.0085
Evaluation 0.1	1000	0.0560	0.0032
	5000	0.0523	0.0056
Evaluation 0.5	1000	0.0623	0.0171
	5000	0.0620	0.0088
Evaluation 1.0	1000	0.0622	0.0130
	5000	0.0598	2.0066

Table 3: Median and the Standard Deviation of the MSE on the Test Partition of the voestalpine Dataset

Sample selection strategy	Population size	Median (MSE)	StDev (MSE)
Generation 0.1	1000	162.17	53.65
	5000	168.49	38.34
Generation 0.5	1000	169.83	58.68
	5000	158.09	49.99
Generation 1.0	1000	178.63	67.41
	5000	189.69	59.44
Evaluation 0.1	1000	198.03	90.25
	5000	184.79	61.48
Evaluation 0.5	1000	169.08	67.04
	5000	183.01	44.45
Evaluation 1.0	1000	160.42	71.94
	5000	159.75	47.77

The results on the voestalpine dataset are shown in Table 3. On this dataset runs that use more samples for training have a slightly lower median MSE. However, the differences between the MSEs are not significant, when compared with the standard deviation.

5. CONCLUSIONS AND OUTLOOK

Not surprisingly a reduction of the evaluated samples decreases the execution time of the GP runs. In addition the result section also shows that runs which use fewer training samples perform as good as runs that use all the available training samples to learn the models. Therefore it is better to use fewer evaluated samples, because the saving of the execution time enables the user to do more test runs or to integrate more advanced, time extensive concepts in the GP algorithm.

Another interesting approach to virtually reducing the number of training samples is using sliding windows in combination with GP (Winkler, Affenzeller and Wagner 2007). An advantage of this technique is that it can be used to predict time-dependent features, which is not possibly with the adaptations described in this paper.

The next step is the integration of an automatically adaption of the relative number of evaluated samples parameter during the algorithm run to improve the achieved qualities while minimizing the necessary execution time. Another interesting approach is stated in the work of Gathercole and Ross (1994). They suggest weighting every sample according to its age (how long the sample has not been used for training) and its difficulty to be predicted correctly. They showed that this method outperforms random subset selection on classification problems and it would be interesting if their method could be adapted for symbolic regression problems.

ACKNOWLEDGMENTS

The work described in this paper was done within the Josef Ressel Centre for Heuristic Optimization *Heureka!* (http://heureka.heuristiclab.com/) sponsored by the Austrian Research Promotion Agency (FFG).

REFERENCES

Durstenfeld, R., 1964, Algorithm 235: Random permutation, *Communications of the ACM*, 7 (7), pp. 420

Gathercole, C., Ross, P., 1994, Dynamic Training Subset Selection for Supervised Learning in Genetic Programming, *Parallel problem solving from nature III*, pp 312- 321, 09-14 October, Jerusalem, Israel

Knuth, D., 1997, *The Art of Computer Programming Volume 2 Seminumerical Algorithms*¸ 3rd Edition, Addison-Wesley Professional, pp 142 - 148

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

Kronberger, G., Feilmayr, C., Kommenda, M., Winkler, S., Affenzeller, M., Bürgler T., 2009, System Identification of Blast Furnace Processes with Genetic Programming, *Proceedings of the IEEE 2nd International Symposium on Logistics and Industrial Informatics (Lindi 2009)*, pp. 63-68, September 10-11, Linz, austria

Poli, R., Langdon, W.B., McPhee, N.F., 2008, *A field guide to genetic programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, (with contributions by J. R. Koza).

Wagner, S., 2009, *Heuristic Optimization Software Systems - Modeling of Heuristic Optimization Algorithms in the HeuristicLab Software Environment*. Thesis (PhD), Johannes Kepler University Linz, Austria.

Winkler, S. M., Affenzeller, M., Wagner, S., 2007, Selection Pressure Driven Sliding Window Behavior in Genetic Programming Based Structure Identification, *Computer Aided Systems Theory – EUROCAST 2007*, pp. 788-795, February 12-16, Las Palmas de Gran Canaria, Spain.

AUTHORS BIOGRAPHIES

MICHAEL KOMMENDA finished his studies in bioinformatics at Upper Austria University of Applied Sciences in 2007. Currently he is a research associate at the UAS Research Center Hagenberg working on data-based modeling algorithms for complex systems within *Heureka!*.

GABRIEL KRONBERGER is a research associate at the UAS Research Center Hagenberg. His research interests include genetic programming, machine learning, and data mining and knowledge discovery. Currently he works on practical applications of data-based modeling methods for complex systems within Josef Ressel Center *Heureka!*.

MICHAEL AFFENZELLER has published several papers, journal articles and books dealing with theoretical and practical aspects of evolutionary computation, genetic algorithms, and meta-heuristics in general. In 2001 he received his PhD in engineering sciences and in 2004 he received his habilitation in applied systems engineering, both from the Johannes Kepler University of Linz, Austria. Michael Affenzeller is professor at the Upper Austria University of Applied Sciences, Campus Hagenberg, and head of the Josef Ressel Center *Heureka!* at Hagenberg.

STEPHAN M. WINKLER received his MSc in computer science in 2004 and his PhD in engineering sciences in 2008, both from Johannes Kepler University (JKU) Linz, Austria. His research interests include genetic programming, nonlinear model identification and machine learning. Since 2009, Dr. Winkler is professor at the Department for Medical and Bioinformatics at the Upper Austria University of Applied Sciences, Campus Hagenberg.

CHRISTOPH FEILMAYR finished his diploma studies in chemical engineering at Technical University of Vienna in 2003. From that time he has worked as research engineer at voestalpine and has been mainly engaged with projects dealing with blast furnace iron making.

STEFAN WAGNER received his MSc in computer science in 2004 and his PhD in engineering sciences in 2009, both from Johannes Kepler University (JKU) Linz, Austria; he is professor at the Upper Austrian University of Applied Sciences (Campus Hagenberg). Dr. Wagner's research interests include evolutionary computation and heuristic optimization, theory and application of genetic algorithms, machine learning and software development.