SIGMA: Systematic Island Grammar forMation Approach

Isaac Griffith and Rosetta Roberts
Empirical Software Engineering Laboratory
Informatics and Computer Science
Idaho State University
Pocatello, Idaho, 83208
Email: {grifisaa@isu.edu, robepea2@isu.edu}

***Abstract*—Introduction:**
Objective:
Methods:
Results: What are the main findings? Practical implications?
Limitations: What are the weaknesses of this research?
Conclusions: What is the conclusion?

***Index Terms*—Island Grammars, Automated Grammar Formation, Software Language Engineering**

I. Introduction

The primary paradigm of software application development has been moving steadily away from the homogenous single language and single technology approach and towards a heterogenous multilingual and multi-technology approach for sometime now [@http://zotero.org/users/1776655/items/B8D2E2AR]. Simultaneosly the use of static and dynamic analysis tools as a standard part of a developer's and organizations toolkit has seen rapid growth. Yet, such tools such as SpotBugs[1], PMD[2], Parasoft dotTEST[3], Pylint[4], etc. have been designed with a single language or family of languages in mind. Services and software issue accumulation tools such as SonarQube[5] have shown that there is a need for static and dynamic analysis tools to be capable of crossing the boundaries of languages. The typical design paradigm for such tools involves the inclusion of multiple languages via plugins (or some other technique) which requires the use of one or more (language specific) parsers to extract knowledge from the code. Unfortunately, this causes two main issues: 1.) A maintenance nightmare for the inclusion and integration of multiple parsers. 2.) A maintenance headache as languages continue to update and change.

The overarching goal of this research (as formatted using the GQM goal template [@http://zotero.org/users/1776655/items/EDCA3974]) is: *Analyze the proposed automated island grammar generation approach with the purpose of evaluation with respect to grammar readability, usability, and maintainability from the point of view of software language engineering researchers and software engineering researchers in the context of developing static and dynamic analysis tools.*

[1]https://spotbugs.github.io
[2]https://pmd.github.io
[3]https://www.parasoft.com/products/dottest
[4]https://www.pylint.org
[5]https://www.sonarqube.org

Organization

II. Background and Related Work

This section details prior studies and work related to the research documented herein. This section is further subdivided into four sections: 1.) the underlying theory concerning Island Grammars and similar concepts, 2.) prior studies concerning the application and evaluation of island grammars, 3.) alternative approaches to the problem noted in Sec. **??** and 4.) the contributions presented in this work.

A. Theoretical Foundations

Island Grammars: Before we formally define these problems, we will borrow the definition of an island grammar from Moonen [@http://zotero.org/users/1776655/items/H26DIBU9].

> "An *island grammar* is a grammar that consists of detailed productions describing certain constructs of interest (the *islands*) and literal productions that catch the remainder (the *water*)."

Moonen [@http://zotero.org/users/1776655/items/H26DIBU9] then formally defined this concept as:

> "Given a language L_o, a context free grammar $G = (V, \Sigma, P, S)$ such that $L(G) = L_o$ and a set of constructs of interest $I \subset \Sigma$ such that $\forall_{i \in I} : \exists_{s_1, s_2 \in \Sigma} : s_1 i s_2 \in L(G)$."

Tolerant Grammars:

Grammar Metrics and Measures: The MCC and HAL metrics were define as part of the grammar metric suites initially defined by Power and Malloy [@http://zotero.org/users/1776655/items/U7GTMISU; @http://zotero.org/users/1776655/items/59S2DPJS].

B. Related Studies

- Island Grammar Studies
- Tolerant Grammar Studies
- Grammar Measure Studies

C. Research Contributions

In this paper we develop the following contributions:

1. The proposal of an algorithmic approach to the generation of island grammars from multiple grammars.
2. Case study results concerning island grammar generation and grammar measurement.
3. A tool which provides the capability to automate the proposed processes.

III. Multilingual Island Grammar Generation

This section describes the methods we have developed to automate the generation of multilingual island grammars. In the following subsections we detail the algorithms we have designed to complete this task. We then describe the operation of these algorithms on simple grammars. Finally, we detail the design of the tool which implements these algorithms and provides a platform to facilitate data collection associated with experimentation with these algorithms.

A. Algorithms

This subsection details the algorithms used in our approach to construct and evaluate multilingual island grammars. Herein we detail three algorithms: the first is defines the approach to constructing multi-lingual island grammars, the second a naive approach to combining grammars (simply for comparison), and the third defines a method to generate sets of interest components. We begin with our multilingual island grammar generation algorithm.

Grammar Meta-Model: The generation of multilingual island grammars begins with the loading of grammars into memory. Due to the fact that their are multiple formalisms for describing grammars and even more file formats available, we have developed a unifying model to represent grammars in memory. This model is depicted in Fig. 1. The model centers around the concept of a grammar which is the composition of a set of productions representing the non-terminal symbols and their derivation rules. Each production is then a composition of rules, where rules represent the body of a production. Each rule also represents the different options for deriving a non-terminal symbol in the grammar. Rules are then an aggregation of a set of symbols, which can either be terminal, non-terminal, or symbol groups. Non-terminal and Terminal symbols are managed via an implementation of the Flyweight design pattern [@http://zotero.org/users/1776655/items/458QBQAK]. Symbol groups are can then either represent a repetition of a set of alternations or a set of optional alternations. An alternation is then a set of symbols.

Surrounding these basic grammar concepts are the utilities used to manipulate and measure the grammar. These include the GrammarLoader which is used to load an individual grammar from its file representation into memory. This is achieved using a parser and parser listener for the given grammar representation language. Finally, controlling all of this is the CommandLineInterface which handles execution of the system as well as interpreting the arguments fed to the program via the Main class and the command line.

Algorithm 1 Multilingual Island Grammar Generation (MIGG)

```
1:  procedure INJECT($persist$, $extern$, $efferent$) // Mark each Interest item in productions of $\mathcal{G}$
2:      for all $i \in \mathcal{I}$ do
3:      end for// Create fresh start, water, and island productions in $\mathcal{G}'$
4:      $\mathcal{S} \longrightarrow (\mathcal{W}\mathcal{I}+)+$
5:      $\mathcal{G}' \longleftarrow (\mathcal{V}, \Sigma, \mathcal{P}, \mathcal{S})$ // Construct Island Production, $\mathcal{I}$
6:      for all $p \in$MARKED($\mathcal{G}$) do
7:      end for// Construct Water Production, $\mathcal{W}$
8:      for all $p \notin$MARKED($\mathcal{G}$) do
9:      end for
10:     $\mathcal{W} \longleftarrow$ REDUCEANDMERGE($\mathcal{W}$)
11:     return $\mathcal{G}'$
12: end procedure
```

Algorithm 2 Naive Island Grammar Generation

```
1: procedure NAIVEGEN($\mathcal{G}$)
       ▷ Initilize components of Island Grammar to be combinations of components of input grammars
2:     $V \longleftarrow \bigcup_{g \in \mathcal{G}} V_g$
3:     $\Sigma \longleftarrow \bigcup_{g \in \mathcal{G}} \Sigma_g$
4:     $P \longleftarrow \bigcup_{g \in \mathcal{G}} P_g$
       ▷ Create fresh start production as alternation among input grammar starts
5:     $S \longleftarrow \bigcup_{g \in \mathcal{G}} S_g$
       ▷ Construct and return the grammar
6:     $G' \longleftarrow (V, \Sigma, P, S)$
7:     return $G'$
8: end procedure
```

Multilingual Island Grammar Generation:

Naive Island Grammar Generation: The experiment detailed in Sec. **??** is designed to evaluate the results of applying the previous algorithm to a grammar. In order to evaluated its effect we are in need of a control treatment which acts as if nothing was effectively applied. In this case, we need some means by which we can combine multiple grammars together into a single grammar, but leave them intact. Thus, we have developed what we call the *Naive Island Grammar Generation* algorithm, depicted in Alg. 2. This algorithm is quite simple in its operation. In lines 2–4 of the algorithm the three set components, V, Σ, P are constructed via a union across the same components of each of the grammars to be combined. In line 5 of the algorithm, a start production is constructed for the new grammar and its body is the unioned set of start symbols from all combined grammars separated by alternation symbols. Finally, in lines 6 and 7 the new grammar is constructed as a tuple of initial components and then returned as the results of the algorithm.

Interest Configuration Generation: Key to the creation of an Island Grammar is the set of interested components. As part of the design of the approach to generate island grammars we

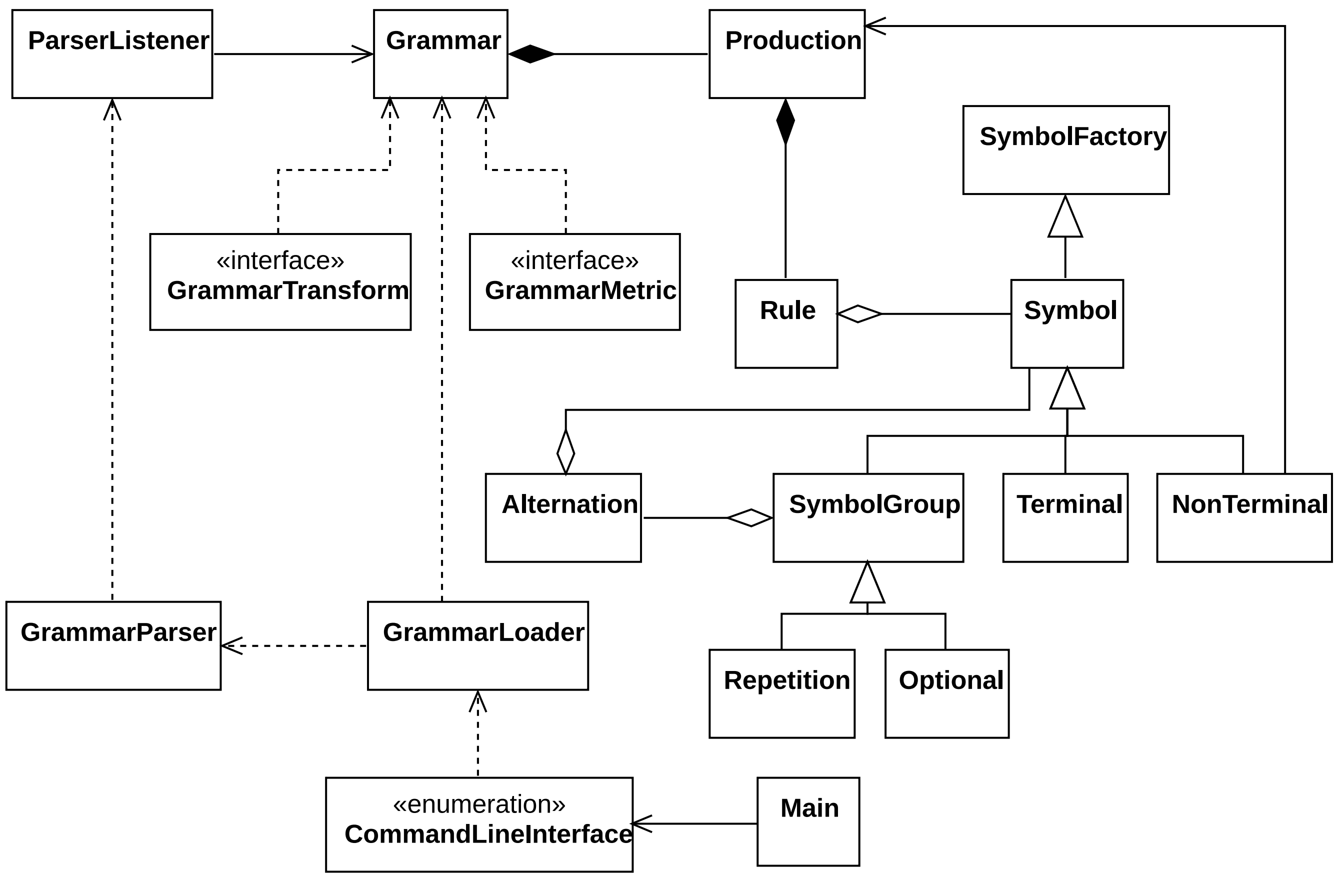

Fig. 1. Metamodel for grammars from [@http://zotero.org/users/1776655/items/62I6AWPB].

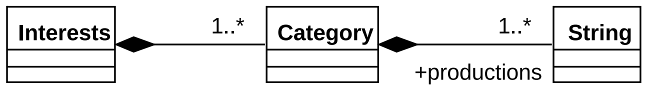

Fig. 2. Interest Configuration Model

Algorithm 3 Interest Configuration Generation

```
1: procedure GENERATE($G$, $n$)
2:     $P \longleftarrow []$
3:     $P \longleftarrow G.\Sigma / G.S$
4:     SHUFFLE($P$)
5:     return $P[1..n]$
6: end procedure
```

developed a model to represent these sets. This also included the definition of an XML schema to represent these concepts outside of the code. The underlying model for these is depicted in Fig. 2.

As part of the experiment detailed in Sec. **??** we need the ability to generate interest configurations for each grammar. We have developed a simple algorithm, depicted in Alg. 3, to do just that. The algorithm takes two parameters G the initial grammar and n the number of interest components to extract. Using this information, the algorithm constructs an empty list, P which is then filled with the non-terminal symbols of the grammar. This list is then randomly shuffled and the top n are selected and returned.

B. Example

C. Tool Design

IV. Experimental Design

This section describes the study design for the experiment used to evaluate the multilingual island grammar generation approach from Sec. III. In developing this study we followed the guidance of Runeson [@http://zotero.org/users/1776655/items/IIF2B7Z7] and Yin [@http://zotero.org/users/1776655/items/KQJDZFIR]. The following subsections detail the research questions, case selection criteria, data collection procedures, and analysis procedures used.

A. Goals, Hypotheses, and Variables

In the spirit of the GQM [@http://zotero.org/users/1776655/items/EDCA3974], we have further refined this goal into a series of directly answerable questions and their underlying rationale. The questions are as follows:

RQ1: How can we automate the combination of languages into a single island grammar?
Rationale:

RQ2: What is the comparative readability of merged grammars?
Rationale:

RQ3: What is the comparative usability of merged grammars?
Rationale:

RQ4: What is the comparative maintainability of merged grammars?
Rationale:

To facilitate answering these questions we have also defined a series of metrics, as follows:

M1: McCabe Cyclomatic Complexity (MCC) a measure of the complexity of a grammar $\mathcal{G}$.
Rationale: A higher complexity represents difficulty in comprehension of the grammar and difficulty in maintaining the grammar. An island grammar is expected to have a lower complexity value than their input grammars.

M2: Halstead Effort (HAL) a measure of effort necessary to maintain grammar $\mathcal{G}$.
Rationale: Similar to MCC, but provides a means by which the MCC is effectively relativised. Island grammars are expected to have a lower complexity and thus lower HAL than their input grammars.

M3: Size of the Set of Input Grammars (Size(G)) the cardinality of the grammars used as input for the island grammar generation.
Rationale: The size of the input grammar set is a key factor in the overall complexity of the generated island grammar and thus should be accounted for.

M4: Size of the Set of Interest Components per grammar (Size(I)) the cardinality of the interest component set used for each grammar acting as input to the island grammar generation algorithms.
Rationale: The size of the interest component set for each grammar used as input for an island grammar is a key factor in the overall complexity of the generated island grammar and thus should be accounted for.

M5: $version(\mathcal{G})$ indicative of the version of the language the grammar $\mathcal{G}$ recognizes. This is a qualitative measure with text data.
Rationale:

The following describes the data to be collected, the data collection process, and how this data is to be stored. First, we describe the data that must be collected. For each grammar under study, we extract the Grammar name, language version, Grammar type, readability score, usability score, and maintainability score for each grammar measured from the Grammar Measure Database, as collected by the tool presented in Sec. III. The data is then accumulated into a table, similar to the example shown in Tab. I, with the following specifications:

- Each row of the represents data associated with a specific grammar set.
- The first column is the grammar set identifier.
- The second column represents the method for island grammar generation.
- The third column represents size of the grammar set.
- The fourth column represents the size of the interest set of each grammar in the set.
- The fifth column represents the calculated complexity of the generated island grammar.
- The sixth column represents the average complexity of the grammar set.
- The seventh column represents the calculated effort of the generated island grammar.
- The eighth column represents the average effort of the grammar set.

The data that is collected then breaks down into the following *independent* and *dependent* variable sets. The independent variables representing the properties of concern for the grammar sets are: Technique, Size(G), and Size(I). The levels associated with technique are *Naive* and *MIGG* depending on the assignment of the set to either the Naive or MIGG approaches for island grammar generation. The levels associated with Size(G) are 2, 5, or 10 which represent a sample of the possible number of grammars to be combined. Finally, the levels associated with Size(I) are 2, 5, and 10 which represent a sample of the possible number of interest components that could be selected from a particular grammar. The dependent variables of concern are ΔMCC and ΔHAL. ΔMCC is the difference between the MCC of the generated grammar and the Average MCC of the source grammars. ΔHAL is the difference between the HAL of the generated grammar and the Average HAL of the source grammars.

Based on the research questions posed above and the variables utilized to evaluate the grammars, we have constructed the following sets of hypotheses. The first set is based on evaluating the change in complexity of the grammars when reduced to a combined Island Grammar. *We expect that when a selection of languages are combined into an Island Grammar with a set of interests imposed, the generated language should be less complex than the average complexity of languages in the input set.* The statistical hypotheses used to test are:

$H_{1,0}$: An increase in the number of grammars shows a decrease in average ΔMCC

$H_{2,0}$: An increase in the number of interests shows a decrease in average ΔMCC

$H_{3,0}$: There is a difference in average ΔMCC between the techniques used to generate island grammars, when controlling for $Size(G)$ and $Size(I)$

The second set is based on evaluating the change in effort

TABLE I
EXAMPLE DATA TABLE FOR THIS STUDY.

Grammar Set	Method	Size(G)	Size(1)	MCC_{IG}	MCC_{avg}	HAL_{IG}	HAL_{avg}
1	Naive	2	2	0.95	0.95	0.95	0.75
1	MIGG	2	5	0.75	0.95	0.95	0.925

to maintain the grammars when reduced to a combined Island Grammar. *We expect that when a selection of languages are combined into an Island Grammar with a set of interests imposed, the generated language should less effort to maintain than the average effort of languages in the input set.* The statistical hypotheses used to test this are:

$H_{4,0}$: An increase in the number of grammars shows a decrease in the average ΔHAL

$H_{5,0}$: An increase in the number of interests shows a decrease in the average ΔHAL

$H_{6,0}$: There is a difference in average ΔHAL between the techniques used to generate island grammars, when controlling for $Size(G)$ and $Size(I)$

B. Design

To gain the most from this experiment, we have elected to use a factorial design. Thus, since we have 2 factors with 3 levels and 1 factor with 2 levels, each replication will need 36 sets of grammars. Furthermore, because the Size(G) variable is based on the number of grammars that are included in the set, we thus need 12 sets at each level of Size(G). Due to this, we need 204 grammars per replication. To gain the most from this experiment we will conduct 3 replications of the experiment, meaning that we require 612 individual grammars.

C. Experimental Units

This section describes the methods of subject sampling and group allocation. For the purposes of this experiment the experimental subjects are grammars representing software languages. Furthermore, as we are concerned with evaluating MIGG from the perspective of creating static and dynamic analysis tools, we have imposed the following restrictions on the types of grammars we are interested in:

- A Grammar may be considered a viable case, if it contains a minimum of 15 different productions.
  Rationale: the grammar should be more than a simple toy grammar.
- A Grammar may be considered a viable case, if it defines a known and oft used programming language.
  Rationale:

With these restrictions in mind we decided to utilize the Grammar Zoo [@http://zotero.org/users/1776655/items/2728FAY2] collection as initial pool from which to select our grammars. Grammar Zoo (at the time of this writing) contains 1029 entries consisting of grammars for a variety of programming languages. With the exception of a few languages most in the collection represent languages currently used in practice by Software Engineers (e.g., Java, Python, Haskell, Dart, etc.).

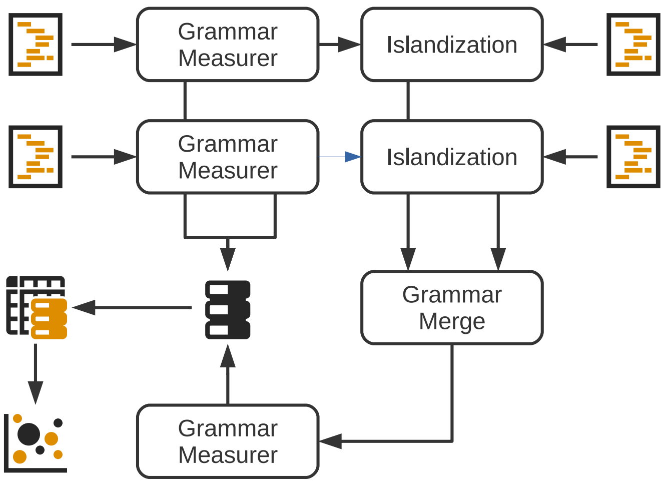

Fig. 3. Data collection process for evaluating multilingual grammar islandization process.

As noted in Sec. IV-B we will need 612 individual grammars to conduct this experiment. These grammars are selected as follows: Initially, 604 grammars are randomly selected from the initial 1029 grammars in Grammar Zoo using a simple random sampling approach. This set is then randomly partitioned into 3 subsets of 104 grammars representing each of the individual replications of the experiment. Within each replication, the grammars are randomly assigned to treatment groups based on the Size(G) characteristic. For each of the treatment groups, a Size(I) level is assigned. Based on this value, a new Interest Configuration for each of the languages is generated using the Interest Configuration generation algorithm presented in Sec. III. Finally, each treatment group is randomly assigned a generation approach for use in the evaluation process.

D. Instrumentation

As part of the experimental design measures for metrics M1–M5 must be collected for each generated island grammar. To do this, the tool described in Sec. III-C provides this capability via implementations of the GrammarMetric depicted in Fig. 1. During program operation, these measures are stored in a database and are retrieve and added to a data table for processing in the analysis phase of the experiment.

E. Data Collection Procedures

In this study, evaluating the effects of islandization and merging process to develop multilingual island grammars we use the process depicted in Fig. 3. This process follows the path identified in Fig. 3 by the numbers encircled in orange, as follows:

1. Initially each grammar is loaded by the Grammar Loader and its data is extracted into the meta model depicted in Fig. 1.
2. A lattice of all combinations of grammars is generated to evaluate the groupings of languages into island grammars.
3. The quality metrics from the quality model are measured on the grammar, and
4. stored in the database.
5. The grammar is then transformed using the *islandization*.
6. This requires that the interest configuration for that grammar be loaded.
7. The generated island grammar is then measured using the same metrics from step 2.
8. For the current combination approach, the set of grammars are then merged to generate a multilingual island grammar.
9. This grammar is then measured, using the same approach as in step 2, and the data is stored to the database.
10. For each grouping defined in the current partitioning of grammars, the IQR of the individual island grammars is measured, and stored to the database.
11. For the partitioning the average IQR is then calculated and stored to the database. This process restarts at step 1, until there are no partitions left.
12. Once all partitions have been processed, the data for all partitions is extracted into a data table ready for analysis.

F. Analysis Procedures

The following analysis procedures are used in answering the questions posed in Sec. **??**. In the first part of this study we have naturally paired samples of data. Given this, we are able to utilize very rudimentary analyses to answer the questions.

For questions RQ3 – RQ5
The results will be in the form of confidence intervals defined at the $\alpha = 97.5\%$ confidence level.

1) Exploratory Data Analysis: To understand the data collected during experimental operation we used a number of basic statistical analyses. Specifically, to understand the dispersion and typical values of the dependent variables we evaluated the mean and standard deviations of ΔMCC and ΔHAL. Furthermore, to understand the nature of the relationships between the dependent and independent variables we evaluated the correlations between all variables in the experiment. Furthermore, we can graphically display all of this information in a lattice structure.

2) Primary Analyses: The primary analyses used in the experiment are designed to evaluate the hypotheses identified in Sec. **??**. The experimental design is a factorial design and thus would be typically evaluated using an ANOVA model. But, ANOVA has several assumptions which if not met leads to a higher chance of Type-II error. To evaluate these assumptions we will be utilizing the following tests:

- To evaluate the normality of the data we will use the Shapiro-Wilk test [@http://zotero.org/users/1776655/items/XDSBUJ53] for normality.
- To evaluate the homogeneity of variance we will use Levene's test for homogeneity of variance [@http://zotero.org/users/1776655/items/QUGJ536S].
- Given the experimental approach we know that the samples are drawn independently and sampled randomly.

In the event that the assumption of the homogeneity of variance is violated but the data is normally distributed, we will apply a weighted variance approach to correct the model. On the other hand, if both assumptions are violated, we will simply return to utilizing the nonparametric Wilcoxson Rank-Sum [] variant of the test. For both experiments evaluating HAL and MCC we will use the following model.

$$y_{ijkl} = \mu + \tau_{ijk} + \epsilon_{ijkl}$$

Where, y_{ijkl} is the observed value of HAL or MCC for the given treatment combination of {ijk} and error {l}. μ is the base line mean value of the MCC or HAL. τ_{ijk} is the treatment value for the i^{th} level of $Method$, the j^{th} level of $Size(G)$ and k^{th} level of $Size(I)$. Finally, ϵ_{ijkl} is the error term representing random error to be controlled for. In the case of both of these experiments we are concerned with determining whether there is any difference due to different treatments, which leads to the following statistical tests:

$\mathrm{H}_{1,0}$: There is no difference in mean ΔMCC
$\mathrm{H}_{2,0}$: There is no difference in mean ΔHAL
$\mathrm{H}_{1,\mathrm{A}}$: There is at least one difference in ΔMCC
$\mathrm{H}_{2,\mathrm{A}}$: There is at least one difference in ΔHAL

In both cases ANOVA and the Rank-Sum based approach simple detect if there is a difference in the means, but does not inform us as to which variable lead to this difference. Thus, in order to detect the difference we will be applying the Tukey-HSV [@http://zotero.org/users/1776655/items/64U6XBSU] analysis to find difference in all means. Additionally, to better understand the effects of individual variables on ΔMCC and ΔHAL we will use the following

G. Evaluation of Validity

Conclusion Validity:
Internal Validity:
Construct Validity:
External Validity:

V. Threats to Validity

This section describes the limitations and threats to the validity of this study. Specifically, we focus on threats to conclusion validity, internal validity, construct validity, content validity, external validity, and reliability in accordance with the framework proposed by Campbell and Cook [@http://zotero.org/users/1776655/items/UX227PF2], Campbell and Stanley [@http://zotero.org/users/1776655/items/8K87M9K7], Wohlin et al. [@http://zotero.org/users/1776655/items/28TSS6TK] and the insights of Yin [@http://zotero.org/users/1776655/items/KQ5D2EFR] and Runeson et al. [@http://zotero.org/users/1776655/items/IIF2B7Z7]

1) Conclusion Validity: Conclusion validity is concerned with establishing statistical significance between the explanatory and response variables. As discussed Section 14.6 the significance of the findings from the presented models meets our expectations. Thus, there are no threats to conclusion validity.

2) Internal Validity: This validity check is concerned with the ability to show a causal relationship between outcomes and treatments. Given that this was a case study focused on the observation of grime relationships and which did not use random selection or random assignment, we cannot statistically show a causal relationship between grime size (GSZ) and pattern type or pattern size.

3) Construct Validity: This validity check is concerned with the use of correct operational measures for the concepts being studied. We utilized detection techniques based directly on the definitions of each form of grime presented in Chapter 9 and by Schanz and Izurieta [43]. We also utilized techniques based directly on the definitions of the relationships for their detection as well. Therefore, there are no threats to construct validity.

4) Content Validity: Content validity is concerned with how well the selected measures cover the content domain. The defined measures of grammar quality were selected based on the limited studies conducted in grammar measurement. Currently, there are no well developed quality models for grammars, which indicates that the selected measures may in fact not cover the entirety of the selected domain. Furthermore, quality is an inherently subjective concept and without expert guidance or well founded justification for measure selection, we cannot be assured that the selected measures adequately cover the content domain. These both point towards potential threats to content validity, but this is mitigated by the underlying justifications of the metrics upon which the quality measures are based. Furthermore, the model is derived from an approach already well founded in the software engineering literature.

5) External Validity: This validity check is concerned with the ability to generalize the results of a study. This study was conducted across multiple grammars describing a variety of programming languages. Unfortunately, as the study considered only those grammars available on GrammarZoo we are unable to extend our results beyond these openly available grammars. However, through the use of random selection of our cases along with the use of multiple contexts, we are not limited to simply the cases evaluated and can advance exploratory inferences.

VI. Conclusions and Future Work

The timeline to complete this study is as follows:

1. Setup tool pipeline: December 15 - January 31
2. Collect Data: February 1 - February 28
3. Analyze Pattern Instances: March 1 - March 31
4. Complete Paper: April 1 - April 30
5. Submit Abstract: 2rd Week of April
6. Publish Results: 3rd Week of April

We intend to publish these results at one of the following conferences:

- ESEM 2020

Acknowledgements

References

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Languages and Systems*. Addison-Wesley, 1994.