Specification Clones: An empirical study of the structure of Event-B specifications

Marie Farrell, Rosemary Monahan & James F. Power

Principles of Programming Research Group,
Maynooth University, National University of Ireland

mfarrell@cs.nuim.ie

International Conference on
Software Engineering and Formal Methods
September 7th, 2017

Maynooth University
National University of Ireland Maynooth

IRISH RESEARCH COUNCIL
An Chomhairle um Thaighde in Éirinn

Outline

Formal Specification in Event-B

- System-level modelling and verification
- Combining event-based logic with basic set theory
- Support for refinement
- Industrial-Strength
- Used for proving safety of a specification
- Proof Support provided by *The Rodin Platform*, an Eclipse IDE for Event-B [1]

Machines

- Model dynamic parts of a system's specification
- Variables, variants, invariants and events
- Events trigger when their guard evaluates to true

```
1  MACHINE mac1
2  VARIABLES  cars_go, peds_go
3  INVARIANTS
4     inv1: cars_go ∈ BOOL
5     inv2: peds_go ∈ BOOL
6     inv3: ¬ (peds_go = true ∧ cars_go = true)
7  EVENTS
8    Initialisation ordinary
9      then  act1: cars_go := false
10           act2: peds_go := false
11   Event set_peds_go ≙ ordinary
12     when  grd1: cars_go = false
13     then  act1: peds_go := true
14   Event set_peds_stop ≙ ordinary
15     then act1: peds_go := false
16   Event set_cars_go ≙ ordinary
17     when  grd1: peds_go = false
18     then  act1: cars_go := true
19   Event set_cars_stop ≙ ordinary
20     then act1: cars_go := false
21 END
```

Contexts

- Model static parts of a systems specification
- Sets, constants and axioms
- Specify new data types to be included in a machine

```
1 CONTEXT ctx1
2 SETS COLOURS
3 CONSTANTS red, green, orange
4 AXIOMS
5     axm1: partition(COLOURS, {red}, {green}, {orange})
6 END
```

Refinement

- Gluing invariants for data refinement
- Superposition refinement
- New events can be added

```
1  MACHINE mac2 refines mac1 SEES ctx1
2  VARIABLES cars_colour, peds_colour,
3  buttonpushed
4  INVARIANTS
5     inv1: peds_colour ∈ {red, green}
6     inv2: (peds_go = true) ⇔ (peds_colour = green)
7     inv3: cars_colour ∈ {red, green}
8     inv4: (cars_go = true) ⇔ (cars_colour = green)
9     inv5: buttonpushed ∈ BOOL
10 EVENTS
11   Initialisation ordinary
12     then act1: cars_colour := red
13          act2: peds_colour := red
14   Event set_peds_green ≙ ordinary
15   refines set_peds_go
16     when grd1: cars_colour = red
17          grd2: buttonpushed = true
18     then act1: peds_colour := green
19          act2: buttonpushed := false
20   Event set_peds_red ≙ ordinary
21   refines set_peds_stop
22     then act1: peds_colour := red
23   Event set_cars_green ≙ ordinary
24   refines set_cars_go
25     when grd1: peds_colour = red
26     then act1: cars_colour := green
27   Event set_cars_red ≙ ordinary
28   refines set_cars_stop
29     then act1: cars_colour := red
30   Event press_button ≙ ordinary
31     then act1: buttonpushed := true
32 END
```

Rodin plugins for Modularising Specifications

Plugin Name	Description
Feature Composition	Composition of Event-B machines and contexts and aids the user in resolving conflicts.
Generic Instantiation	Instantiate and reuse generic developments within other formal developments.
Model Decomposition	Decomposition of Event-B machines/contexts using the shared variable and shared event styles.
Pattern	Reuse of existing Event-B models including refinement steps within a development in order to save the modelling and proving effort.
Parallel Composition	Composition of Event-B machines using the shared event approach.
Modularisation	Allows the developer to construct modules and prove modular developments.
Renaming Refactory	Renames Event-B model elements so that the changes are propagated through the relevant machines, contexts and proof obligations.
Theory Extension	Extends the Event-B mathematical language (potentially with new data types) and the Rodin proving infrastructure.

Corpus of Event-B Specifications

We have assembled a corpus of Event-B specifications that has the following composition:

	Total	Larger Projects	Smaller Projects
Projects	85	16	69
Machines	468	219	249
Contexts	359	209	150
Events	10828	9254	1574





We split the corpus into "smaller" and "larger" projects using Tukey's test and trimming on the number of sentences per project (lines of code).

Sources: Event-B wiki, DEPLOY website, ABZ Case Study Tracks, direct from developers.

Metrics

"One accurate measurement is worth more than a thousand expert opinions"

- Grace Hopper

Quantifying Specification Size

All Rodin projects are available as a set of XML files, thus we developed a suite of `Python` programs to analyse them.

Larger Projects								
Project	Macs	Cons	Evs	Refs	Sens	Auto	Inter	RP
Midas*	43	61	2500	40	26395	2034	3163	2183
FlashFileFS	18	6	320	13	5442	974	531	88
DepSatSpec	14	2	2094	13	4771	1309	549	0
ATM	7	12	129	6	3447	925	37	46
B2Bminip	12	0	228	11	2900	425	73	124
Bepiv3.3	6	6	137	1	2665	153	113	12
TSHHDMac	35	50	1487	18	2661	602	84	15
Bepiv5.0	9	10	329	8	2007	683	317	0
CDIS	7	6	103	6	1894	101	0	3
HDMac	19	25	718	16	1605	448	23	2
Pilot_v3	4	4	98	3	1586	134	9	0
MLLanding	11	2	313	10	1432	286	210	0
FlashFileFL	6	12	109	5	1243	379	13	11
HLanding	11	7	321	9	1213	173	68	17
ch3_press	8	3	144	7	1200	0	0	0
OnbCont	9	3	224	8	1108	438	1	14

Summary Statistics

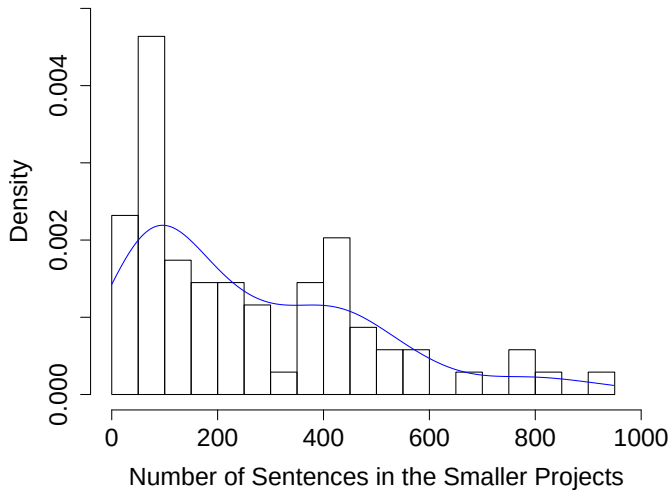
All Projects ($n = 85$)								
Project	Macs	Cons	Evs	Refs	Sens	Auto	Inter	RP
Minimum	0	0	0	0	29	0	0	0
Median	4	2	27	3	274	83	5	8
Maximum	43	61	2500	40	26395	2034	3163	2183
MADN	3.0	1.5	28.2	3.0	298.0	86.0	7.4	11.9

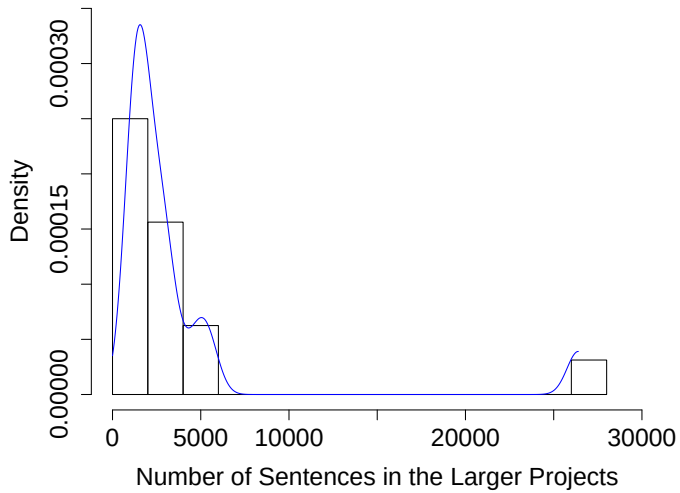
Smaller Projects ($n = 69$)								
Project	Macs	Cons	Evs	Refs	Sens	Auto	Inter	RP
Minimum	0	0	0	0	29	0	0	0
Median	3	2	17	2	192	56	5	8
Maximum	10	10	106	9	948	560	70	47
MADN	1.5	1.5	16.3	1.5	206.1	54.9	7.4	11.9

Larger Projects ($n = 16$)								
Project	Macs	Cons	Evs	Refs	Sens	Auto	Inter	RP
Minimum	4	0	98	1	1108	0	0	0
Median	10	6	270	8	1950	431	70	11
Maximum	43	61	2500	40	26395	2034	3163	21.83
MADN	5.2	5.9	203.9	4.4	1076.4	398.1	97.1	17.0

Sentence Kernel Distributions

- Sentence: axiom, invariant, variant, parameter, guard, witness or action.





Comparisons using Spearman's rank correlation coefficient

The most notable **strong** correlations ($p < 0.001$) were:

- **the number of events and number of sentences** in the small data set ($\rho = 0.905$), where the median number of sentences per event is 11 ($madn = 4.4$). However, in the larger project set, this correlation is weak ($\rho = 0.391$). The larger projects contain a greater number of contexts, thus adding sentences to the projects that are not sentences within events.
- **the number of machines and the number of events** in both the smaller ($\rho = 0.849$) and larger ($\rho = 0.904$) project sets. The median number of events per machine is 25 ($madn = 9.8$) in the larger set and 5 ($madn = 2.7$) in the smaller.

Automatic vs Interactive Proofs

It is important to measure the relative amount of theorem proving work imposed on the user as projects increase in size and complexity.

- 78.6% of the proofs were automatic in the larger projects.
- 91.1% of the proofs were automatic in the smaller projects.

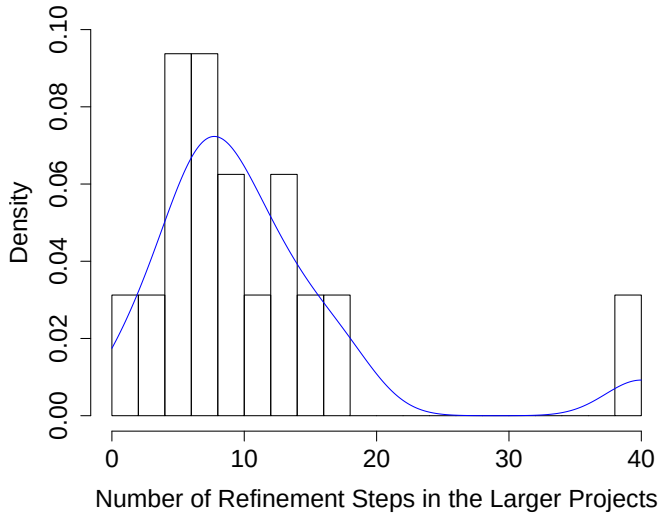
This metric can be used over time to measure the degree to which automated theorem proving has increased in effectiveness.

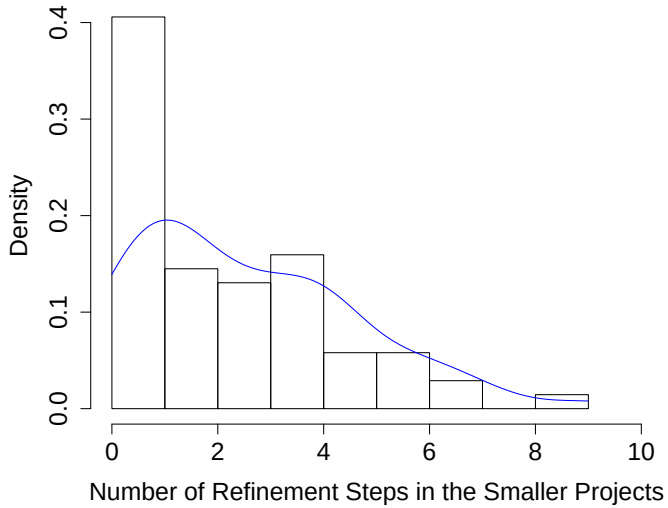
Refinement Kernel Distributions and Correlations

There are strong correlations between:

- **the number of machines and the number of refinement steps** $\rho = 0.989$ and $\rho = 0.993$ respectively ($p < 0.001$).
- **the number of refinement proofs and the number of refinement steps** in a project in the smaller project set ($\rho = 0.786$, $p < 0.001$) resulting in the median ratio of 3 refinement proofs to 1 refinement step.

  This correlation is not significant for the larger project set because developers of the larger projects often opted to avoid data refinement and use event extending to streamline their developments (5/16).





Code Clones

"Cloning may be good and it may be bad. Probably, it is a bit of both."

- Richard Dawkins

Good: Promote the reuse of reliable code and can save time/effort in development.

Bad: Duplicated code can indicate limitations in the programming paradigm's modularisation mechanisms.

Clone Types

Type-1: *Exact* matches (syntactically the same).

Type-2: *Exact* matches modulo renaming of variables, types etc.

Type-3: Type-1 or Type-2 with some insertions or deletions (subset).

Type-4: *Functionally* equivalent [3].

Machines, contexts and events are compared

Parsing: We represent the components of the Rodin files as an abstract syntax tree and then we perform comparisons at context, machine and event level.

Preprocessing:

- Sentences are tokenised to eliminate white space and formatting.
- Sentences are only compared against others of the same type (axioms with axioms etc.)
- Threshold of 2 or less sentences to exclude trivial matches.

Machines, contexts and events are compared

Clone identification:

Type-1: exact matches between the full sentence sequences in each case.

Type-2: matches between the full sentence sequences, but where variable names are anonymised, each variable name being replaced by a positional indicator.

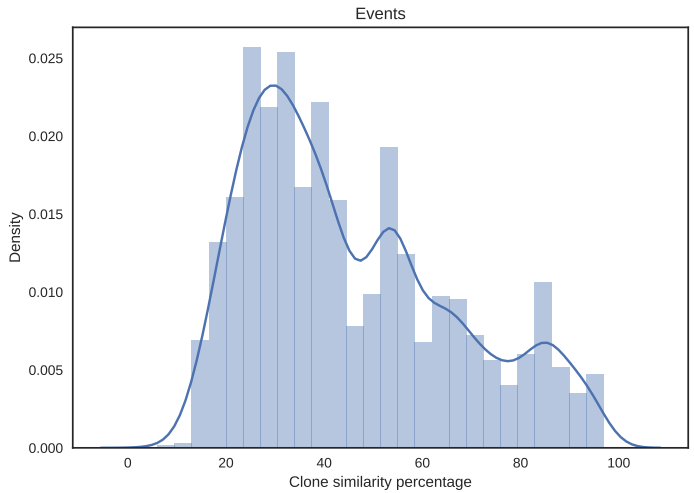
Type-3: matches between two sentence sequences (with variable names anonymised or unanonymised), except that now we allow matches between sub-sequences of the sentences.

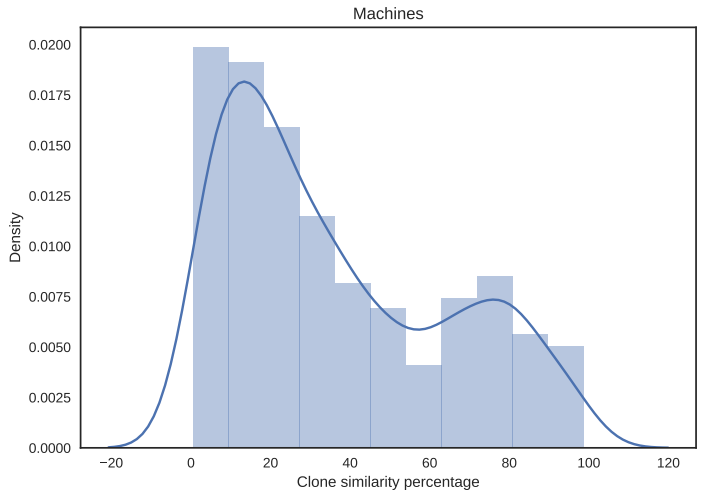
Results

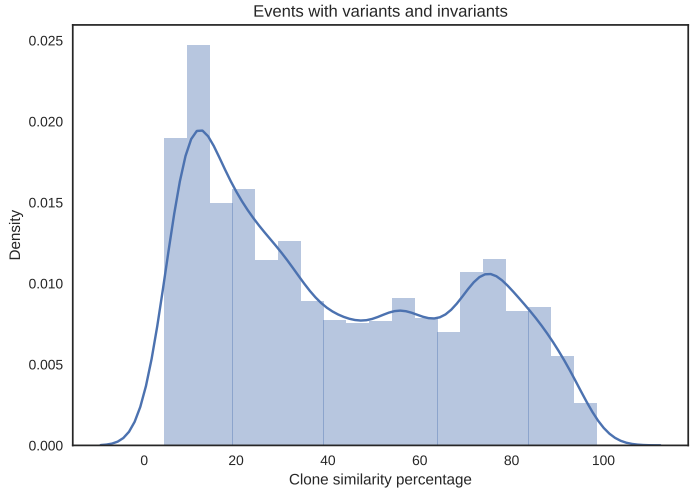
Event-B Component	Clone Pairs				Actual Clones	
	Type-1	Type-2	Type-3	Total	Total	Occur.
Contexts	18	0	22	40	22	51
Machines	13	7	937	957	19	40
Machines (+VI)	9	7	943	959	13	28
Events	276	942	4781	5999	131	417
Events (+VI)	35	158	7229	7422	65	175

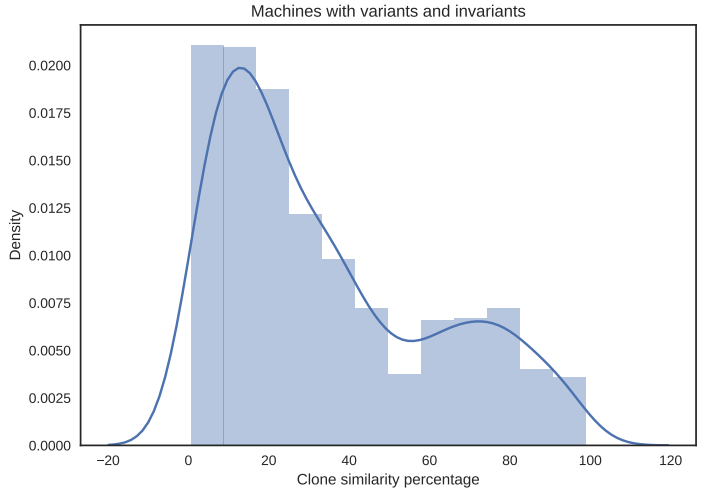
Table: The occurrence of clone pairs and clones per type throughout the entire corpus. Note that '(+VI)' indicates that the variants (where appropriate) and invariants have been included in the analysis.

Distribution of Type-3 Clones









Inter and Intra Project Clonings

Contexts

Of the 22 individual context clones, 18 occurred on an inter project basis indicating that they are being reused in a manner similar to libraries.

Machines

The vast majority of machine clones occurred as intra project clones and therefore were most likely caused by refinement chains.

Events

Intra project clonings occurred 382 times, in fact there were 210 scenarios where one event was a clone of another event in the *same* machine. This accounts for 17.2% of the total type-1 and type-2 event clones.

Modularisation of Event-B Specifications

There are a number of ways that we could reduce the number of clones:

- Use any of the Rodin plugins for modularisation that we outlined earlier.
- Use the specification-building operators made available in our institution for Event-B [2].

Threats to Validity

- The selection of our corpus poses a threat to **conclusion validity** since we are dealing with a heterogeneous group of projects, and there is a risk that the differences in metrics are due to other factors not measured here.
- While our defined and measured metrics corresponded to major syntactic categories in Event-B and have clear analogies with similar constructs in programming languages, there is a threat to **construct validity** here.
- In adapting the definition of code clones to Event-B we made a number of decisions on what should be measured and the degree of matching involved; altering these could yield different results.
- Our corpus may not be a large enough sample size to study thus causing a threat to **external validity** in terms of the generalisability of our results.

Our Contributions

- Application of existing software engineering approaches of calulating metrics and detecting code clones to specifications written in Event-B.
- We have provided and analysed the metrics of a corpus of Event-B specifications and thus contributed a benchmark against which other Event-B developments can gauge their comparative size and complexity level.
- Our empirical study supports the claim that there is an underlying requirement for modularisation constructs in Event-B.

Future Work

- The assessment of *clone genealogies*, particularly in the context of refinement - how do clones evolve throughout successive refinements.
- Detecting non-typing invariant clones would allow us to analyse data refinement clones using gluing invariants.
- We did not assess the size of the state (number of variables per machine) in this study but we intend to address this as future work. This will allow us to investigate whether or not the size of the state is indicative of the complexity of the system being modelled.

References


J.-R. Abrial.
Modeling in Event-B: System and Software Engineering.
Cambridge University Press, New York, NY, USA, 1st edition, 2010.


M. Farrell, R. Monahan, and J. F. Power.
Providing a Semantics and Modularisation Constructs for Event-B using Institutions.
In *International Workshop on Algebraic Development Techniques*, 2016.


C. K. Roy, M. F. Zibran, and R. Koschke.
The vision of software clone management: past, present, and future.
In *Software Maintenance, Reengineering and Reverse Engineering*, pages 18–33, Antwerp, Belgium, 2014.

Questions?

http://www.cs.nuim.ie/~mfarrell