Metamorphic Relation Template v1.0

Sergio Segura, Amador Durán, Javier Troya and Antonio Ruiz-Cortés
{sergiosegura,amador,jtroya,aruiz}@us.es

Applied Software Engineering Research Group
University of Seville, Spain
January 2017

Technical Report ISA-17-TR-01

This report was prepared by the

Applied Software Engineering Research Group (ISA)
Department of computer languages and systems
Av/ Reina Mercedes S/N, 41012 Seville, Spain
http://www.isa.us.es/

Support: This work has been partially supported by the European Commission (FEDER), by Spanish Government under CICYT project BELI (TIN2015-70560-R) and the Andalusian Government project COPAS (P12-TIC-1867).

List of changes

Version	Date	Description
1.0	January 2017	First release

Metamorphic Relation Template v1.0

Sergio Segura, Amador Durán, Javier Troya and Antonio Ruiz Cortés
Department of Computer Languages and Systems
University of Seville, Seville, Spain
{sergiosegura,amador,jtroya,aruiz}@us.es

Abstract

Metamorphic testing enables the generation of test cases in the absence of an oracle by exploiting relations among different executions of the program under test, called metamorphic relations. In a recent survey, we observed a great variability in the way metamorphic relations are described, typically in an informal manner using natural language. We noticed that the lack of a standard mechanism to describe metamorphic relations often makes them hard to read and understand, which hinders the widespread adoption of the technique. To address this shortcoming, we have proposed a template for the description of metamorphic relations, which aims to ease communication among practitioners as well as contributing to research dissemination. Also, it provides a helpful guide for those approaching metamorphic testing for the first time. This technical report describes the proposed template, records its evolution through its different versions and shows several examples of use.

Index Terms

Metamorphic testing, metamorphic relation, templates

I. Introduction

Metamorhpic testing enables the generation of test cases when the expected output of a program execution is complex or unknown [1], [2]. To that purpose, rather than checking the correctness of each individual program output, metamorphic testing checks whether multiple executions of the program under test fulfill certain conditions referred to as *metamorphic relations*. A metamorphic relation is a necessary property of the program under test that relates two or more input data and their expected outputs, e.g. $\sin(x) = \sin(-x)$. In the last two decades, hundreds of metamorphic relations have been reported in a variety of domains including web services and applications [3], computer graphics [4], compilers [5], machine learning [6] and cybersecurity [7].

In a recent survey, some of the authors reviewed 119 papers on metamorphic testing published in the last two decades [8]. We observed that most metamorphic relations are informally described using natural language, which may lead to misunderstandings and communication problems among researchers and practitioners. We also found that key information about the relations was often omitted or simply assumed to be known by the reader. Additionally, we found a great variability in the way metamorphic relations are described, which makes them hard to read and understand. We think that this variability could be explained by the degree of expertise on the technique. We observed that experienced researchers tend to clearly identify metamorphic relations including helpful data as identifiers, preconditions or examples. Conversely, newcomers on the technique usually describe the relations informally as a part of the main research text, omitting key information like a precise definition of the program's inputs and outputs. Finally, some authors have proposed to use formal notations to describe metamorphic relations, but their approach have not been widely adopted probably due to the difficulty to be understood by all stakeholders [9].

The problem of capturing and expressing information in a way that it is understandable for users with different degree of expertise has been addressed in fields such as requirements engineering [10], experimentation [11] and software metrics [12], [13]. A classical approach to address this problem is the use of templates. A *template* is a combination of placeholders and linguistic formulas used to describe something in a particular domain, e.g. an experiment. Templates facilitate communication among practitioners, contribute to research dissemination, and provide a helpful guide for beginners.

This technical report presents, in detail, a template-based approach for describing metamorphic relations. The proposed template is based on the structure of metamorphic relations observed in the literature, and it is also inspired by related and widely adopted templates in various fields of software engineering [10], [11], [13]. The template is intentionally simple and flexible to foster its adoption by the metamorphic testing community. To this purpose, the template specifies *what* data must be included in the description of a metamorphic relation, but not *how* it must be specified, allowing the use of natural language, formal languages or a combination of both. This document describes the template and its evolution (e.g., when feedback from other researchers is received) through version control. The proposed template is used to define several previously published metamorphic relations from different domains and groups of authors, showing that the template is expressive enough to represent all the subject relations.

In the following, the template is presented. Then, several examples of metamorphic relations published in the literature are described with it.

II. Template

The template for describing metamorphic relations is shown below, where the placeholders are depicted between < and > and optional sections are enclosed between square brackets.

In the domain of <*application domain*>
[**where** <*context definition*>]
[**assuming that** <*constraints*>]
the following metamorphic relation(s) should hold

- <*metamorphic relation name*$_1$>:
  if <*relation on inputs/outputs*>
  then <*relation on inputs/outputs*>

  ...
- <*metamorphic relation name*$_n$>:
  if <*relation on inputs/outputs*>
  then <*relation on inputs/outputs*>

The template placeholders have the following meaning:

application domain
: This is the application domain in which the metamorphic relations apply. For example: general domains such as search engines, code obfuscators or machine learning; specific versions of software tools such as Weka 2.1; software services like Google search, etc.

context definition
: The context definition includes all necessary definitions of concepts, variables, notations, etc. used in the definition of the metamorphic relations and that are essential for their proper understanding. The section containing this placeholder is considered as optional because depending on the complexity of the metamorphic relations and the degree of formalization, could not be strictly necessary.

constraints
: In this optional section, some constraints can be specified indicating necessary conditions for the metamorphic relation to be applicable.

metamorphic relation name
: This is the name of the metamorphic relation being defined. Ideally, it could be a meaningful name, but a simple label is also acceptable in order to distinguish it from other metamorphic relations defined in the same template.

relation on inputs & outputs
: These are logical implications in which both the antecedent (the *if* placeholder) and the consequent (the *then* placeholder) are relations defined over the function inputs and outputs.

III. Examples of use

This section shows several examples of metamorphic relations taken from the literature and expressed with our template. More specifically, we selected 10 metamorphic testing papers, from 35 different authors and 8 different application domains, from which 17 metamorphic relations were selected to be described using our approach. We may remark that these relations were randomly selected with the only purpose of having a representative pull of metamorphic relations, and not because we identified any specific limitations in them. Table I depicts the list of selected papers including publication year, short list of authors, title, application domain, and reference. Five of the papers were published in journals, and five in conferences or workshops. In all the examples, we tried to follow as much as possible the names and the definition style used by the original authors.

Year	Authors	Title	Domain	Ref.
2002	T.Y. Chen et al	Metamorphic Testing of Programs on Partial Differential Equations: a Case Study	Numerical programs	[14]
2004	T.H. Tse et al	Testing Context-Sensitive Middleware-Based Software Applications	Embedded systems	[15]
2009	W.K. Chan et al	Finding failures from passed test cases: improving the pattern classification approach to the testing of mesh simplification programs	Computer graphics	[4]
2010	K.Y. Sim et al	Detecting Faults in Technical Indicator Computations for Financial Market Analysis	Financial software	[16]
2010	X. Xie et al	Testing and validating machine learning classifiers by metamorphic testing	Machine learning	[6]
2011	F.-C. Kuo et al	Testing Embedded Software by Metamorphic Testing: a Wireless Metering System Case Study	Embedded software	[17]
2014	S. Segura et al	Automated metamorphic testing of variability analysis tools	Software variability	[18]
2015	Z.Q. Zhou et al	Metamorphic Testing for Software Quality Assessment: A Study of Search Engines	Search database	[3]
2016	T.Y. Chen et al	Metamorphic Testing for Cybersecurity	Cybersecurity	[7]
2016	M. Lindvall et al	Agile Metamorphic Model-based Testing	Search database	[19]

Table I
SELECTED PAPERS

A. Metamorphic relation in [14]

In the domain of *thermodynamics*
where

- G_i is a mesh grid of $n \times n$ positions
- P is a point on the plate
- $T_{G_i}(P)$ is the temperature at point P determined using the mesh grid G_i
- G_i ($n \times n$ mesh grid) $\subset$ G_j ($m \times m$ mesh grid) if $n < m$

the following metamorphic relation(s) should hold

- R_{PDE}:
  if $G_i \subset G_j \subset G_k$
  then $T_{G_i}(P) \leq min\{T_{G_j}(P), T_{G_k}(P)\} \vee T_{G_i}(P) \geq max\{T_{G_j}(P), T_{G_k}(P)\}$.

B. Metamorphic relation in [15]

In the domain of *smart streetlight systems*
where

- p_i represents the position at point i
- $r^2(p_i, p_0)$ is a function to return the square of the distance between the streetlight at position p_0 and the visitor at position p_i
- l_{n_i} is the illumination (radiance) at the visitor site (p_i)
- l_{f_i} is the favorite illumination (radiance) of the visitor at position p_i
- ε denotes a tolerance threshold
- r_{eff} is the radius of the effective illumination region of a streetlight
- the symbol $\approx$ denotes that two values are approximately equal within an application-specific tolerance limit of 2ε

the following metamorphic relation(s) should hold

- $MR_{PowerUp}$:
  if $r^2(p_1, p_0) \leq r_{eff}^2 \wedge r^2(p_2, p_0) \leq r_{eff}^2 \wedge l_{f_1} = l_{f_2}$
  then $l_{n_1} \approx l_{n_2}$.

C. Metamorphic relations in [4]

In the domain of mesh simplification programs
where

- u is a function accepting an image and returning an outline of a shape in the image
- m is a 3D polygonal model composed of a sequence of vertices $\langle v_1, v_2, ..., v_(n-1), v_n \rangle$
- each vertex of a 3D polygonal model is composed of three coordinates ($coor_x, coor_y, coor_z$)
- P is a program
- $P(m)$ is an image produced by a program P over an input m
- $\subseteq_c$ is a two-polygon containment relation, which asserts that the left-hand side should be within the right-hand side
- *flip* takes an image and puts it upside down, i.e., it flips it vertically

the following metamorphic relation(s) should hold

- MR_1:
  if
  noScale is a function that accepts a 3D polygonal model and returns the 3D polygonal model with simplification percentage being 100
  then $u(P(m)) \subseteq_c u(P(noScale(m)))$.
- MR_2:
  if *reverse* is a standard sequence reversal function that accepts a sequence $\langle v_1, v_2, ..., v_{n-1}, v_n \rangle$ and returns the reversed sequence $\langle v_n, v_{n-1}, ..., v_2, v_1 \rangle$
  then $P(m) = P(reverse(m))$.
- MR_3:
  if *yInvert* is a function that accepts a 3D polygonal model and performs the y-coordinate transformation ($coor'_y = -coor_y$) over the sequences of vertices in the model
  then $P(m) = flip(P(yInvert(m)))$.

D. Metamorphic relations in [16]

In the domain of technical indicators for financial market analysis
where

- $P(p_k, p_{k-1}, p_{k-2}, \ldots, p_0)$ are the price values from time period $t = k$ to $t = 0$.
- $SMA(n, t)$ is the simple moving average value for n consecutive time periods from t (inclusive).

the following metamorphic relation(s) should hold

- *MR1 of Simple Moving Average (SMA)*:
  if $p_t > p_{t+n}$
  then $SMA(n, t) > SMA(n, t+1)$.
- *MR4 of Smoothed Moving Average (SMMA)*:
  if $p_t < p_{t+1}$ AND $n1 > n2$
  then $SMMA(n2, t+1)$–$SMMA(n2, t) > SMMA(n1, t+1)$–$SMMA(n1, t)$.

E. Metamorphic relations in [6]

In the domain of machine learning classifiers
where

- S is the training data set.
- t_s is a source test case, i.e., a data sample $\langle a_0, a_1 \ldots a_{m-1}. \rangle$
- l_i is the class label obtained as the output of t_s.
- an uninformative attribute is one that is equally associated with each class label.

the following metamorphic relation(s) should hold

- *MR-2.1 Addition of uninformative attributes*:
  if in the follow-up input, an uninformative attribute is added to each sample in S and to t_s
  then the output of the follow-up test case should still be l_i.
- *MR-5.1 Removal of classes*:
  if in the follow-up input, we remove one entire class of samples in S of which the label is not l_i
  then the output of the follow-up test case should still be l_i.

F. Metamorphic relation in [17]

In the domain of wireless signal metering
where

- P_{ant1} and P_{ant2} are different signal powers to antenna
- ΔP_{ant} is the difference between P_{ant1} and P_{ant2}
- $\Delta RSSI$ is the difference between the computed received signal strength indicators corresponding to P_{ant1} and P_{ant2}

the following metamorphic relation(s) should hold

- *Meter reading function*:
  if P_{ant1} and P_{ant2} are within the range $[-100 \text{ dBm}, -70 \text{ dBm}]$
  then ΔP_{ant} must be within the range $[1.5\Delta RSSI - 3, 1.5\Delta RSSI + 3]$.

G. Metamorphic relation in [18]

In the domain of feature model analysis tools
where
- M is a feature model.
- $\Pi(M)$ is a function returning the set of products of a feature model M.
- $\#$ is the cardinality function on sets.

the following metamorphic relation(s) should hold
- $MR_1 Mandatory$:
  if M' is derived from M by adding a mandatory feature f_m as a child feature of f_p
  then $\#\Pi(M') = \#\Pi(M) \ \wedge$
  $\forall p \in \Pi(M) \ \bullet \ f_p \notin p \ \Rightarrow \ p \in \Pi(M') \ \wedge f_p \in p \ \Rightarrow \ (\ p \cup \{f_m\} \ ) \in \Pi(M')$

H. Metamorphic relations in [3]

In the domain of Google search
where
- *site:* is a Google search operator that specifies domains, e.g. `site:nbc.com`
- q_1 and q_2 are queries represented as sequences of conjunctive search criteria, i.e. $q_i = \langle \ c_j \ \rangle_{j=1..n}$
- an exact word or phrase is a search criterion, e.g. "`side effect of antibiotics in babies`"
- site:d is also a search criterion
- $R(q)$ is the result set of web pages returned by a given query q, i.e. $R(q) = \{ \ p_k \ \}_{k=1..m}$
- $\#R(q)$ is the size of $R(q)$
- $\frown$ is the sequence concatenation operator
- *rev* is the reverse sequence function, i.e. $q = \langle \ c_j \ \rangle_{j=1..n} \ \Rightarrow \ rev(q) = \langle \ c_j \ \rangle_{j=n..1}$

assuming that
- $0 < \#R(q_1) \leq 20$

the following metamorphic relation(s) should hold
- *MPSite*:
  if $q_2 = q_1 \frown$ site:d where d is the domain of one of the web pages in $R(q_1)$
  then $R(q_2) \subseteq R(q_1)$, i.e. the results of q_2 must be a subset of the results of q_1
- *MPReverseJD*:
  if $q_2 = rev(q_1)$, i.e. q_2 is the reverse of q_1
  then $R(q_2) \approx R(q_1)$, i.e. the results of q_2 are similar to the results of q_1 applying Jaccard similarity.

I. Metamorphic relations in [7]

In the domain of code obfuscators
where
- p, p_1 and p_2 are computer programs
- Ω is a program obsfuscation function
- $\Omega(p)@[t_i]$ is the obsfuscation of p at a given time t_i
- $\equiv$ is the program functional equivalence relation

the following metamorphic relation(s) should hold
- MR_1:
  if $p_1 \ \equiv \ p_2$, i.e. p_1 and p_2 are functionally equivalent
  then $\Omega(p_1) \ \equiv \ \Omega(p_2)$, i.e. the obsfucations of p_1 and p_2 are also functionally equivalent.
- MR_2:
  if $\{ \ t_i \ \}_{i=1..n}$ are different times
  then $\forall \, i : 1..n-1 \ \bullet \ \Omega(p)@[t_i] \equiv \ \Omega(p)@[t_{i+1}]$, i.e. the obfuscation process does not depend on the obfuscator environment (time of execution in this case).

J. Metamorphic relations in [19]

In the domain of the NASA's Data Access Toolkit (DAT)
where

- $q_1, q_2, \ldots, q_n$ with $n \geq 2$ are identical search queries for the DAT system.
- $R(q_i)$ is the set of results of a given query q_i.

the following metamorphic relation(s) should hold

- MR_{order}:
  if the order of the parameters of the queries is changed
  then $R(q_1) = R(q_2) = \ldots = R(q_n)$.
- $MR_{Tformat}$:
  if the time values of the queries are changed to a different format, but representing the same time lapse
  then $R(q_1) = R(q_2) = \ldots = R(q_n)$.

REFERENCES

[1] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Technical Report HKUST-CS98-01, Department of Computer Science, The Hong Kong University of Science and Technology, Tech. Rep., 1998.

[2] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Fault-based testing without the need of oracles," *Information & Software Technology*, vol. 45, no. 1, pp. 1–9, 2003. [Online]. Available: http://dx.doi.org/10.1016/S0950-5849(02)00129-5

[3] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 264–284, March 2016.

[4] W. K. Chan, J. C. F. Ho, and T. H. Tse, "Finding failures from passed test cases: Improving the pattern classification approach to the testing of mesh simplification programs," *Software Testing, Verification and Reliability Journal*, vol. 20, no. 2, pp. 89–120, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1002/stvr.v20:2

[5] V. Le, M. Afshari, and Z. Su, "Compiler validation via equivalence modulo inputs," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '14. New York, NY, USA: ACM, 2014, pp. 216–226. [Online]. Available: http://doi.acm.org/10.1145/2594291.2594334

[6] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *The Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, Apr. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2010.11.920

[7] T. Y. Chen, F. C. Kuo, W. Ma, W. Susilo, D. Towey, J. Voas, and Z. Q. Zhou, "Metamorphic testing for cybersecurity," *Computer*, vol. 49, no. 6, pp. 48–55, June 2016.

[8] S. Segura, G. Fraser, A. Sanchez, and A. Ruiz-Cortes, "A survey on metamorphic testing," *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, Sept 2016.

[9] Z. Hui and S. Huang, "A formal model for metamorphic relation decomposition," in *Fourth World Congress on Software Engineering (WCSE), 2013*, Dec 2013, pp. 64–68.

[10] A. Durán, B. Bernárdez, A. Ruiz-Cortés, and M. Toro, "A requirements elicitation approach based in templates and patters," in *2nd. Workshop on Requirements Engineering (WER)*, Buenos Aires, Argentina, Sep 1999, p. 17–29.

[11] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer-Verlag Berlin Heidelberg, 2012.

[12] V. R. Basili, "Software modeling and measurement: The goal/question/metric paradigm," College Park, MD, USA, Tech. Rep., 1992.

[13] R. van Solingen and E. Berghout, *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*. McGraw-Hill, 1999. [Online]. Available: https://books.google.co.uk/books?id=EczdPAAACAAJ

[14] T. Y. Chen, J. Feng, and T. H. Tse, "Metamorphic testing of programs on partial differential equations: A case study," in *Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*, ser. COMPSAC '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 327–333. [Online]. Available: http://dl.acm.org/citation.cfm?id=645984.675903

[15] T. H. Tse, S. S. Yau, W. K. Chan, H. Lu, and T. Y. Chen, "Testing context-sensitive middleware-based software applications," in *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, Sept 2004, pp. 458–466 vol.1.

[16] K. Y. Sim, C. S. Low, and F.-C. Kuo, "Detecting faults in technical indicator computations for financial market analysis," in *2nd International Conference on Information Science and Engineering (ICISE), 2010*, Dec 2010, pp. 2749–2754.

[17] F.-C. Kuo, T. Y. Chen, and W. K. Tam, "Testing embedded software by metamorphic testing: A wireless metering system case study," in *IEEE 36th Conference on Local Computer Networks (LCN), 2011*, Oct 2011, pp. 291–294.

[18] S. Segura, A. Durán, A. B. Sánchez, D. L. Berre, E. Lonca, and A. Ruiz-Cortés, "Automated metamorphic testing of variability analysis tools," *Software Testing, Verification and Reliability*, vol. 25, no. 2, pp. 138–163, 2015. [Online]. Available: http://dx.doi.org/10.1002/stvr.1566

[19] M. Lindvall, D. Ganesan, S. Bjorgvinsson, K. Jonsson, H. S. Logason, F. Dietrich, and R. E. Wiegand, "Agile metamorphic model-based testing," in *Proceedings of the 1st International Workshop on Metamorphic Testing*, ser. MET '16. New York, NY, USA: ACM, 2016, pp. 26–32. [Online]. Available: http://doi.acm.org/10.1145/2896971.2896979