Inf Syst Front (2016) 18:819–823
DOI 10.1007/s10796-016-9695-1

Towards common reusable semantics

Thouraya Bouabana-Tebibel[1] · **Stuart H. Rubin**[2]

Published online: 2 September 2016

1 Introduction

Reuse has emerged as a promising field of software and system engineering, where previously acquired knowledge is reused and/or transformed to create new knowledge. Reuse means acquiring model artifacts from another context instead of creating the artifacts from scratch. It involves a process to specify, produce, classify, retrieve and adopt artifacts such as semi-formal/formal models or executable specifications or fragments thereof for the purpose of use in assembly activities. Reuse provides for quality improvement and productivity increases, since developers spend more time testing and validating than writing code. Data rescue and reuse also contribute to predictions of the future. Information and knowledge derived from past trends sustain future predictions and underpin attribution needed to drive policy responses. For example, ecosystems are a predilection domain for such studies. Sustained observations have long been recognized as important in disentangling climate driven change from regional and local-scale anthropogenic impacts and environmental fluctuations or cycles in natural systems (Hawkins et al. 2013; Palczewska et al. 2014).

✉ Thouraya Bouabana-Tebibel
t_tebibel@esi.dz

Stuart H. Rubin
stuart.rubin@navy.mil

[1] LCSI Laboratory, Ecole nationale Supérieure d'Informatique, Algiers, Algeria

[2] Space and Naval Warfare Systems Center Pacific, San Diego, CA 92152-5001, USA

Reuse may be subdivided into planned and unplanned components (Prieto-Diaz 1996; Aldin and De Cesare 2011). Whereas, unplanned reuse does not require any special preparation of a model to enable it, planned reuse can only be achieved by first preparing a model, documenting it, and creating metadata (Koschmider et al. 2014). The costs associated with preparing, adapting, and ensuring that a component is appropriate for reuse in a particular context often outweigh the benefits of unplanned reuse. The process of reuse can also be refined in technical (architecture, framework or repository) and methodological aspects (Koschmider et al. 2014). Methodology is split into: (1) abstraction by encompassing works related to patterns, reference models or metamodels, (2) selection by describing approaches related to retrieval and similarity of models, (3) specialization through works that elaborate on the configuration, customization or adaptation of models, and (4) integration describing the composition of models out of fragments and modules.

Existing literature (Varnell-Sarjeant and Andrews 2015) shows that reuse in embedded systems is, usually less successful than reuse in nonembedded systems. It also appears that some of the difference in outcomes might be related to the artifacts reused; but, few of the empirical studies focus on artifacts and their impact on reuse. Most of them are merely reusing code. Thus, several questions are left to be answered (Varnell-Sarjeant et al. 2015). First, do embedded systems use different development approaches than nonembedded systems? Second, what artifacts can be reused for successful outcomes? Third, do reuse outcomes vary between embedded systems and nonembedded systems? Fourth, does reuse effectiveness vary with project size and developer expectations? Some of these research questions were answered in (Varnell-Sarjeant et al. 2015). Reuse effectiveness does not vary with system type (Q3), however, nuances of the development approaches does vary (Q1). There is a significant difference in

what artifacts are reused (Q2). There is no statistical difference in reuse success based on reporting of the outcome variables. Embedded systems are significantly more likely to reuse hardware, test products, and test clusters.

Artifacts assembly is constrained by formal descriptions of conditions imposed by certain design choices made in the architecture description. The benefit from these constraints is twofold. First, they play a documentation role and thus help developers in understanding an architecture description and the design choices made during its construction. Second, they play the role of invariants that can be checked after an architectural evolution (Tyree and Akerman 2005). However, repetitive constraint specifications, for revised architecture descriptions, is a fastidious task for a developer, whose role is to define high-level component-based architectural descriptions, and who moreover is accustomed to the extensive reuse of component descriptors. Defining such specifications through the reuse of component descriptors is thus advocated. As a result, architectural constraint specifications are decomposed, thus yielding a common repository of reusable, customizable and stable (i.e., already tested and validated) component descriptors for software architects (Tibermacine et al. 2016).

Reuse and integration also intervene within formal specifications, implementations and tools (Bouabana-Tebibel and Rubin 2015; Bouabana-Tebibel and Rubin 2016). In (Barbosa et al. 2016), it is shown how to reuse and integrate different specification logics in an undergraduate course on formal software specification. The proposed hybridization represents asymmetric logic combination in the sense that specific features of hybrid logic are developed on top of another logic. This leads to the design of a new course along which students' progress from equational to hybrid specifications in a uniform setting – integrating paradigms, combining data and behavior, and dealing appropriately with systems evolution and reconfiguration. More generally, formal methods serve as a support for handling reuse in a rigorous way. They are first employed to prove or disprove safety properties of the designed components at early stages of the development process. They are subsequently deployed to check consistency of the assembled components.

Despite these advances, effective practical and formal mechanisms for finding, understanding, and adapting reusable components are currently lacking. Existing approaches typically suffer from high computational costs – thus causing difficulties when applied to real-world applications. New approaches, based on heuristics and randomization, are developed to increase the utility of reuse, while minimizing search and requisite space (Rubin and Bouabana-Tebibel 2015; Rubin and bouabana-Tebibel 2016).

This special issue tackles the theory and formal foundations, which underpin reuse. It incorporates reuse and integration as overt operational procedures, or as operations built into the formal representation of the data and operators employed. It includes nine articles, five of which are expanded versions of the best papers presented at the IEEE 16th International Conference on Information Reuse and Integration and its joint IEEE 3rd International Workshop on Formal Methods Integration, which was held in San Francisco on August 13–15, 2015.

2 Papers in the special issue

Evaluating the semantic suitability of components for new usage, in different scenarios, is a challenging task. One especially grand concern is ranking the identified components in a way that best meets the pragmatic non-functional reuse goals of the user. It is a major issue, which is nevertheless lacking investigation. The main difficulty appears in the quantitative weighting of criteria related to non-functional properties. Existing approaches, at best, only allow developers to specify one non-functional objective at query time; and when they do, it is only possible to specify very general metrics, which reflect neither the developer's exact reuse context nor the complexity involved in adapting a component for reuse. Providing more advanced mechanisms for ranking components based on quality criteria would thus constitute a significant step forward towards component reuse. In (Kessel and Atkinson 2016), the authors propose an effective and reliable method for ranking software components based on their non-functional properties, without requiring users to provide quantitative weighting information. Non-functional criteria, designated by software metrics with only minimal information about their relative importance, are specified.

Next, a fundamental task in reuse is to determine the most appropriate component(s) for the current reuse context. Goal model-based reuse allows modelers to capture the candidate's potential for reuse in terms of its positive and negative impacts on the high-level goals and objectives of stakeholders, which in turn help in reasoning about the most appropriate candidate. Goal models are combined with other models that impose additional constraints on the most appropriate candidate – like feature models and workflow models. In (Duran and Mussbacher 2016), an approach based on goal models is proposed to characterize different candidate solutions according to the impacts they have on high-level stakeholder/system goals. Evaluation of high-level goals, non-functional requirements, system qualities, and candidate solutions is based on relative and quantitative values to determine normalized satisfaction ratings for high-level goals. A proof-of-concept implementation of the novel evaluation mechanism is discussed.

Efficient retrieval of the appropriate information, under difficult scenarios, still attracts researchers from the reuse community. It is not uncommon that some semantic concepts are too difficult to be retrieved accurately. Sometimes, it is due to

the rare presence of these semantic concepts in the training set. Under other circumstances, data sets are imbalanced, with very small positive-to-negative ratios. Thus, the data mining and machine learning-based methods fail to work properly, since they are mostly built based on the assumption that the positive-to-negative ratio is neither too small nor too large. In (Chen et al. 2016), a framework is developed for building positive subspace models on a set of positive training sets. This framework supports the retrieval of semantic concepts from highly imbalanced datasets. A novel weighted-ranking score method fuses the final ranking scores from all positive subspace models. Another major aspect of reuse is component integration, which intervenes as a complementary process. It handles the artifacts analysis, in view of their subsequent synthesis into the new whole. Component assembly is even more delicate when it deals with semantic aspects. The main challenges related to such integration result from overlapping artifacts. Configuration fragments, concerned with the non-functional properties such as availability, security, or the performance of a system may be an illustrative application domain. In (Jahanbanifar et al. 2016), a model-based approach defines a consistent integration of configuration fragments into a system configuration. The authors refer to the model weaving technique to capture the semantics of the relations between the entities representing the configuration fragments. The constraints, corresponding to these semantic relations, are used to guard the configuration consistency during runtime modifications.

Safety plays a crucial role on critical systems; and, it is the responsibility of the safety assessment process. Consistency of reuse is usually checked through empirical experiments and/or formal methods. While the former provide weak validation of reuse, the latter give strong evidence of its correctness. In (Hansen et al. 2016), the authors develop a formal semantic framework for modeling in a core assembly language to capture the latter essential features. Their solution is based on small code fragments, which are independent and emulate a specific hardware instruction in a safe manner. They apply statistical model checking to model, analyze, and quantify program behavior in the presence of fault models. Model-checking is one of the most commonly used verification techniques. It provides a good compromise between reliability of checking and complexity of use. In (Nellen et al. 2016), an approach is presented to reduce the verification effort through a counterexample-guided abstraction refinement method. The refinement is embedded into the reachability analysis process in order to prevent it from repeatedly re-checking the same model behavior. The approach also combines bounded model checking on discrete models with reachability analysis on hybrid models that are restricted to specific control paths. Safety-relevant parts of the complex system dynamics are considered in the verification process. The method is applied to the area of plant control.

Hard formalisms are also investigated. In (Harris 2016), category theory is used as a theoretical foundation for faceted browsing. The author demonstrated that category theory can act as a theoretical foundation for faceted browsing that also encourages reuse and interoperability. Reuse and interoperability appear at two levels: between systems and within a system. The proposed model works at both levels by leveraging category theory as a common language for representation and computation. The author demonstrates how the interactive process and its underlying structures can be mathematically abstracted, and how a consistent view of facets as objects and morphisms between objects can be obtained. Resulting implementations can then be constructed through a category-theoretic lens – allowing for abstract comparisons and communications that naturally support interoperability and reuse. Similarly, In (Didier and Mota 2016), a foundational theory is defined to support a more precise representation of fault events. Fault modeling depends on which analyses are intended to be performed. The failure logic is essential for system safety assessment because it is used as basic input for building fault trees. The authors define an algebra of temporal faults and provide several laws to enable the algebraic reduction of formulas. They generalize the laws in terms of abstract properties. Next, they illustrate the application of the laws on a real case study provided by industry.

The need for heuristics pervades the real world. These are not mere afterthoughts, but serve the goals of randomization itself. As a result, any 5th generation programming system, any knowledge acquisition system, and/or any mechanics for formal representation must embody a heuristic component if it is to be reusable and integrated in a non-trivial way. Rubin et al. (2016) address the question as to whether domain transference and the reuse of problem-solving knowledge can be mediated through the reuse of heuristics; and, if so, the extent to which such transference may occur in the solution of *NP-hard* problems. This paper offers a constructive proof of the unbounded density of knowledge in support of the Semantic Randomization Theorem. It details this result and its potential impact on the machine learning community.

3 Creativity and challenges

Knowledge reuse is sometimes the object of criticism despite its great underpinning benefits. The question arises – does knowledge reuse preserve creativity? In order to survive and flourish, managers do not only need to make decisions quickly, but also need to make them innovatively (Massetti 1996; Marakas and Elam 1997; Mumford 2000; Majchrzak et al. 2004; Rubin 2012).

A capability to generate creative ideas that are both novel and valuable emerges as an essential resource for sustainable competition. An empirical study, conducted in (Cheung et al. 2008),

examines the effects of knowledge reuse on individual creativity outcomes, with particular interest in the potential effects of personal creativity. It shows that creativity does not depend wholly on one's personality. Knowledge reuse does have very different, if not opposite, effects on one's creativity performance. While providing explicit, codified knowledge seems to increase the number of ideas that a creative person can generate, the quality of the resulting ideas may be significantly lowered. Thus, knowledge reuse should be cautiously managed so that individual creativity is not unintentionally suppressed.

The KASER (Rubin et al. 2004) was among the first demonstrations of computational creativity by way of transformative analogy. It showed, in 2004, that a machine could be creative with far greater validity than mere chance would allow. Moreover, it learned to estimate the chance of error in its predictions. Some domains (e.g., mathematics, chemistry, physics, et al.) provided for more analogies than others (e.g., historical dates). Those domains demonstrated that reuse need not be limited to literal recaps; rather, they showed that reuse exists along a taxonomy of abstractions. The more general the abstraction, the greater the utility of the reusable object. Such reuse is not subject to essential incompleteness on account of it being heuristically derived. Rather, by accepting a small, but inherent error creep, intelligent systems of far greater practical utility emerge.

Transformational analogies need not be limited to one step, but may be transitive with cumulative error. So long as this error is managed for the domain, great creativity can emerge. The KASER allows the user to specify how many levels of inductive creativity are allowed. For example, in computational medicine, one might wish to set this number to zero to prevent error; whereas, in computational artwork, one might wish to set this number to five levels of inductive creativity for visual effect. Even the knowledge of how many levels of creativity are to be allowed is subject to creative interpretation. Also, a patient with an incurable disease might be more willing to accept experimental medicine than one for which an adequate cure exists. What we find is that deduction may be valid, but validity is, in general, less valuable than knowledge, which is near perfect – particularly if the inherent error can be bounded.

Constructs that are closely related are said to be *symmetric* (e.g., hot and cold); whereas, constructs that are unrelated or only related by a long chain of transforms are said to be *random* (e.g., hot and vacuum). Most constructs contain elements of randomness and symmetry. New knowledge can be created through the reuse of both in transformative analogy. Such knowledge is often open under deduction, not provable, and invaluable for problem-solving in the form of heuristics. The KASER (Rubin et al. 2004) demonstrated that transformative reuse is more powerful than logical deduction. Perhaps the successor to the Japanese Fifth Generation Project will be based on transformative analogy.

In trying to equate syntax to derive a common semantics (e.g., "Hello how are you?" and "Hi how is it going?"), the question arises as to how to measure similarity. Again, the answer is randomization (Rubin et al. 2004). By mapping each phrase to a minimal, or theoretically-speaking near-minimal form, the phrases may be equated – allowing for slight degrees of fuzzy mismatch, which are inherent in general. Thus, randomization may be expected to play the underpinning role in the semantic web and all forms of natural language query. It is evidenced that natural language understanding is every bit as creative as computational analogy – it's just a slightly different domain.

In matching a context against a rule base, the context is randomized into several minimal forms. This is necessary because, in theory, the minimal form may be unattainable and/or non-deterministic. These minimal forms are compared against a situational rule base to select a rule to fire, as is true for a KASER. This serves to mitigate the impedance mismatch between humans and intelligent machines. It opens the door for ever-more natural communications between humans and machines. Thus, it would appear that the essence of our entire being – from use of natural language, creative thoughts, visual and auditory recognition, and even sensory fusion is fundamentally dependent on transformative analogy. This is in stark contrast to early effort to mimic human capabilities using the predicate calculus. It provides hope for the future and a recipe for progress.

Acknowledgments We are very grateful to all referees for providing constructive critiques of the contributed papers. We would like to acknowledge them for their high quality reviews and pertinent comments. Many thanks to: Antonio Badia (USA), Bedir Tekinerdogan (Netherlands), Daniel Neagu (USA), David Spivak (USA), Djamel Eddine Menacer (Algeria), Djamel Eddine Saidouni (Algeria), Dorina Petriu (Canada), Eray Tüzün (Turkey), Frank Zeyd (UK), Habiba Drias (Algeria), Kais Klai (France), Krzysztof Kepa (USA), Kyungmin Bae (USA), Lawrence Chung (USA), Luca Bernardinello (Italy), Mariofanna Milanova (USA), Martin Walker (UK), Michele Loreti (Italy), Nadjet Kamel (Algeria), Nguena-Timo Omer (Canada), Osman Hasan (Pakistan), Pablo Sanchez (Spain), Paul Vickers (UK), Ruben Gran (Spain), Salah Sadou (France), Sebastian Goetz (Germany), Sergiy Bogomolov (Austria), Sidi Mohamed Benslimane (Algeria), Slimane Larabi (Algeria), Stéphane Jean (France), Sven Overhage (Germany), Syed Rafay Hasan (USA), Thao Dang (France), Victoria Yoon (USA), Vincent Aravantinos (Germany), Walid Khaled Hodouci (Algeria), Yahya Slimani (Tunisia).

References

Aldin, L., & De Cesare, S. (2011). A literature review on business process modelling: new frontiers of reusability. *Enterprise Information Systems, 5*(3), 359–383.

Barbosa, L. S., Martins, M., Madeira, A., & Neves, R. (2016). Reuse and integration of specification logics: the hybridisation perspective. *Theoretical information reuse and integration. Advances in Intelligent Systems and Computing, 446*, 1–30.

Bouabana-Tebibel, T., & Rubin, S. H. (2015). Formalisms for reuse and systems integration. *Advances in Intelligent Systems and Computing, 346*. doi:10.1007/978-3-319-16577-6.

Bouabana-Tebibel, T., & Rubin, S. H. (2016). Theoretical information reuse and integration. *Advances in Intelligent Systems and Computing, 446*. doi:10.1007/978-3-319-31311-5.

Chen, C., Shyu, M.-L., & Chen, S.-C. (2016). Weighted subspace modeling for semantic concept retrieval using gaussian mixture models. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796–016-9660-z.

Cheung, P.-K., Chau, P. Y. K., & Au, A. K. K. (2008). Does knowledge reuse make a creative person more creative? *Decision Support Systems, 45*, 219–227.

Didier, A., & Mota, A. (2016). An algebra of temporal faults. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796–016–9664-8.

Duran, M. B., & Mussbacher, G. (2016). Investigation of feature run-time conflicts on goal model-based reuse. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796-016-9657-7.

Hansen, R. R., Larsen, K. G., Olesen, M. C., & Wognsen, E. R. (2016). Formal modelling and analysis of bitflips in ARM assembly code. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796–016–9665-7.

Harris, D. R. (2016). Foundations of reusable and interoperable facet models using category theory. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796-016-9658-6.

Hawkins, S. J., Firth, L. B., McHugh, M., Poloczanska, E. S., Herbert, R. J. H., Burrows, M. T., Kendall, M. A., Moore, P. J., Thompson, R. C., Jenkins, S. R., Sims, D. W., Genner, M. J., & Mieszkowska, N. (2013). Data rescue and re-use: recycling old information to address new policy concerns. *Marine Policy, 42*, 91–98.

Jahanbanifar, A., Khendek, F., & Toeroe, M. (2016). Semantic weaving of configuration fragments into a consistent system configuration. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796–016–9663-9.

Kessel, M., & Atkinson, C. (2016). Ranking software components for reuse based on non-functional properties. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796–016–9669-3.

Koschmider, A., Fellmann, M., Schoknecht, A., & Oberweis, A. (2014). Analysis of process model reuse: where are we now, where should we go from here? *Decision Support Systems, 66*, 9–19.

Majchrzak, A., Cooper, L. P., & Neece, O. E. (2004). Knowledge reuse for innovation. *Management Science, 50*(2), 174–188.

Marakas, G. M., & Elam, J. J. (1997). Creativity enhancement in problem solving: through software or process? *Management Science, 43*(8), 1136–1146.

Massetti, B. (1996). An empirical examination of the value of creativity support systems on idea generation. *MIS Quarterly, 20*(1), 83–97.

Mumford, M. D. (2000). Managing creative people: strategies and tactics for innovation. *Human Resource Management Review, 10*(3), 313–351.

Nellen, J., Driessen, K., Neuhäußer, M., Ábrahám, E., & Wolters, B. (2016). Two CEGAR-based approaches for the safety verification of PLC-controlled plants. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796–016–9671-9.

Palczewska, A., Palczewski, J., Robinson, R. M., & Neagu, D. (2014). Interpreting random forest classification models using a feature contribution method. Integration of reusable systems. *Advances in Intelligent Systems and Computing, 263*, 193–218.

Prieto-Diaz, R. (1996). Reuse as a new paradigm for software development. *In Proceedings of the international workshop on systematic reuse: issues in initiating and improving a reuse program* (pp. 1–13). UK: Liverpool.

Rubin, S. H. (2012). On creativity and intelligence in computational systems. *Advances in reasoning-based image processing intelligent systems. Intelligent systems reference. Library, 29*, 383–421.

Rubin, S. H., & Bouabana-Tebibel, T. (2015). NNCS: randomization and informed search for novel naval cyber strategies. *Recent advances in computational intelligence in defense and security. Studies in Computational Intelligence, 621*, 193–223.

Rubin, S. H., & Bouabana-Tebibel, T. (2016). Naval intelligent authentication and support through randomization and transformative search. *New approaches in intelligent control - techniques, methodologies and applications. Intelligent Systems Reference Library, 107*, 73–108.

Rubin, S. H., Murthy, S. N. J., Smith, M. H., & Trajkovic, L. (2004). KASER: knowledge amplification by structured expert randomization. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 34*, 2317–2329.

Rubin, S. H., Bouabana-Tebibel, T., & Hoadjli, Y. (2016). On the empirical justification of theoretical heuristic transference and learning. *Information Systems Frontiers, 18*(5). doi:10.1007/s10796–016-9661-y.

Tibermacine, C., Sadou, S., That, M. T. T., & Dony, C. (2016). Software architecture constraint reuse-by-composition. *Future Generation Computer Systems, 61*, 37–53.

Tyree, J., & Akerman, A. (2005). Architecture decisions: demystifying architecture. *IEEE Softw., 22*(2), 19–27.

Varnell-Sarjeant, J., & Andrews, A. A. (2015). Comparing reuse strategies in different development environments. *Advances in Computers, 97*, 1–47.

Varnell-Sarjeant, J., Andrews, A. A., Lucente, J., & Stefik, A. (2015). Comparing development approaches and reuse strategies: an empirical evaluation of developer views from the aerospace industry. *Information and Software Technology, 61*, 71–92.

Thouraya Bouabana-Tebibel is a professor of computer science at Ecole nationale Supérieure d'Informatique in Algeria. She received a Ph.D. in Computer Science from USTHB University (Algeria) in collaboration with Pierre & Marie Curie University (France) in 2007. She has been an engineer/researcher for eight years. She was the Director of the Doctoral School in Sciences and Technologies of Information and Communication for four years and is now a director of research at Laboratoire de Communication des Systèmes Informatiques. She has over 95 refereed publications and a book edited by Editions Universitaires Européennes. She was co-editor (with S. H. Rubin) of three Springer books in the Advances in Intelligent Systems and Computing series. She has successfully conducted numerous research projects. Her focus areas of interest include formal methods, cyber security, and randomization-based methodologies

Stuart H. Rubin is a senior scientist at the Space and Naval Warfare Systems Center (SSC) in San Diego, code 71,730 (Advanced Concepts & Applied Research). He was previously a tenured associate professor of computer science at Central Michigan University (CMU). He received a Ph.D. in Computer and Information Science from Lehigh University in 1988. He was previously an ONT Post-Doctoral Fellow, at NOSC, for three years. He has over 30 Assigned Navy Patents, over 292 Refereed Publications, and received SSC-PAC's Publication of the Year Awards in 2007, 2009, 2010, and 2011. He was co-editor (with T. Bouabana Tebibel) of three Springer books in the Advances in Intelligent Systems and Computing series. Dr. Rubin is a SIRI Fellow and serves in leadership roles in numerous IEEE technical societies. His focus areas of interest include computational creativity and knowledge-discovery systems – emphasizing heuristic, evolutionary, fuzzy, and randomization-based methodologies.