Measuring the quality of transformation alternatives in software architectures evolution

Javier Criado[1], Silverio Martínez-Fernández[2], David Ameller[2] and Luis Iribarne[1]

[1] Applied Computing Group, University of Almería, Spain
{javi.criado,luis.iribarne}@ual.es

[2] GESSI Research Group, Universitat Politècnica de Catalunya, Barcelona, Spain
{smartinez,dameller}@essi.upc.edu

Abstract. Many today's software systems need to be self-adapted at run-time. Model transformation is a good approach to adapt the component-based architecture of such software systems. However, existing model transformation processes focus on the functionalities of systems, giving less importance to the quality attributes. The goal of this study is to improve model transformation processes by also considering quality attributes in the generation and adaptation of component-based architectures (*i.e.*, driving the selection among many alternative model transformations by software architecture metrics). Such metrics evaluate the qualities of an architecture, such as flexibility and modifiability. This paper provides some measures of quality for different transformation alternatives and an example in the ENIA software.

Keywords: component-based software, architecture configuration, architecture metrics, quality-driven transformation, model transformation.

1 Introduction

Nowadays, component-based software systems need to be self-adapted at run-time. Previous studies have shown that model transformation is a good approach to adapt the component-based architecture of software systems [4]. However, existing model transformation processes focus on the functionalities of systems, giving less importance to the Quality Attributes (QA), also known as non-functional requirements or *-ilities*. An exception are the quality-driven model transformations described in [7], in which quality is introduced on the design of the transformation process, or the proposed mechanisms related to the Dynamic Software Product Lines (DSPLs) [8] considering some quality information.

This paper presents an approach to adapt and evolve component-based software systems by measuring the quality of different transformation alternatives. The goal of this study is to improve model transformation processes by also considering QAs in the generation and adaptation of component-based architectures. As an example, this paper will focus on *flexibility* metrics to show the suitability of our QA-based transformation approach. We provide an exemplar scenario for the ENIA (ENviromental Information Agent) software. ENIA is a geographic information system whose User Interfaces (UI) are based on coarse-grained components and need to be adapted at run-time depending on user interactions, system requirements, or other evolution needs [1].

2 QA-based Transformation Approach

Depending on a system's targeted QAs and goals (*e.g.*, improve its flexibility, maximize the modifiability, minimize the cost, or optimize the execution performance), architectural design decisions can be oriented in different ways. In this sense, decisions about the construction of software architectures, such as component selection, may differ from each other. For this reason, it is important to include QAs as part of the rationale to make such architectural decisions. The use of specific QA metrics allows us to analyze different alternatives of component configurations, and how each one of these alternative architectures is related to the QA considered.

In order to demonstrate the feasibility of this approach, we focus on some metrics related to the scenario of ENIA UIs. Nevertheless, these measurement actions can be extended and applied to other UI applications offering its functionality as a mashup or a dashboard, *e.g.*, Geckoboard, Cyfe, and Netvibes.

In the ENIA scenario, we discussed the relevant QAs: flexibility, modifiability, analyzability, performance, testability and consumed resources. Due to the limitation of space, this paper focuses on the highest priority of ENIA: the *flexibility* attribute [5] of the *quality in use model* of the ISO/IEC 25000 standard [6]. ENIA UIs (intended for managing geographic data, social network information, third-party widgets, etc.) try to provide a friendly interaction by accomplishing the following objectives: (a) user interfaces must be elastic and flexible with the aim of allowing the modification of their structure; (b) users can reconfigure and customize their interfaces, *e.g.*, resizing the component displayed in the interface, changing its position, adding new components, and removing existing ones; and (c) user interfaces with a greater number of simple components are preferred over interfaces with a lower number of complex components gathering a large amount of functionality (*i.e.*, the more pieces has an interface, the more reconfiguration operations can be performed on it).

The metrics addressed to measure the flexibility of ENIA UIs are calculated from the component and architecture specifications (*i.e.*, information provided by developers, architects, and experts) and used at run-time. This paper proposes an initial stage of the work based on two simple derived metrics. Simple and realistic metrics allow easier adoption in industry. The first metric is related to the number of components. The second metric is related to the *isResizable* property of the specification, which indicates whether a component can be resized or not. Table 1 shows the information about both metrics (rc and rr).

The values obtained from these metrics are used for choosing the best alternative of architecture that can be constructed from a previous one. Both values (rc and rr) must be maximized and, therefore, a trade-off must be performed.

This information is intended to be used not only for guiding the design and development of UIs, but also to be applied for supporting the adaptation of this type of architectures at run-time [4].

Figure 1 shows an example of the measurement that can be performed in the context of the ENIA system. Consider the initial UI provided by ENIA prototype implementation [1]: a software architecture representing an UI made up from two components, a map (M) for displaying geographical information layers, and a component showing the messages of a social network account (T). From this UI, if we want to incorporate a legend (L) and a list of layers (LL) related to the map, it is possible to choose among three alternatives. In this example, the number in labels represent different alternatives of the same component type (*e.g.*, M1, M2, and so on). M3 gathers the functionalities of M, L and LL (which is represented by its three provided interfaces). C1 acts as a container delegating the interfaces of M2, L2 and LL2. Moreover, M1, M3, T1, L1, and C1 are resizable whereas M2, L2 and LL2 are not. The values obtained for rc and rr shows that A12 has the best value of rc, but A13 has the best value of rr. Nevertheless, A11 has the best value for the average of rc and rr, 0.775 instead of the 0.7 of A12 and A13.

The transformations to get these alternatives are different. The transformation for adapting A0 to A11 (T1) implies the addition of L1 and LL2 and their connection to M1 whereas A13 is obtained from the replacement of M1 by M3 (*i.e.*, performing T3 transformation). Therefore, if we intend to maximize the

Metric	Description	Derived	Exp.
c	Number of components	n	—
r	Number of resizable components	n	—
m	The highest c from all architectural alternatives	y	$max(c_1, \ldots, c_n)$
rc	Ratio of components according to m	y	$rc = c/m$
rr	Ratio of resizable components	y	$rr = r/c$

Table 1. Example metrics for maximizing the flexibility in ENIA user interfaces

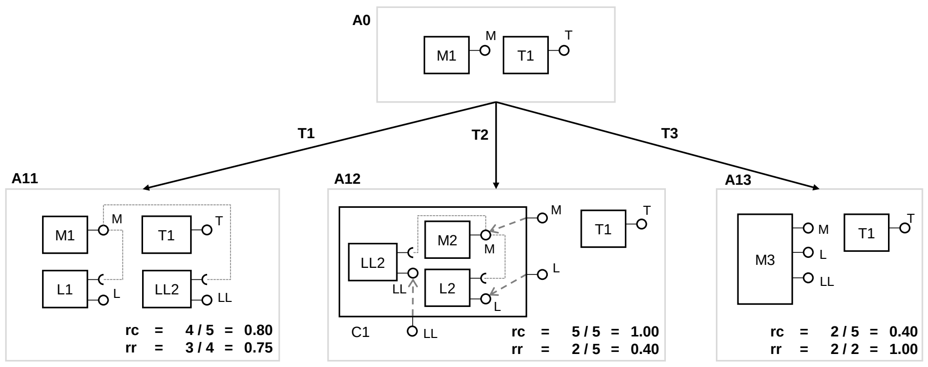

Fig. 1. Example of transformation alternatives

flexibility, T1 can be chosen as the best transformation operation for adapting the architecture of this example scenario.

3 Further Considerations

It is well accepted in the software architecture community that QAs are the most important drivers of architecture design [2]. Therefore, QAs should guide the selection of the best transformation alternative, considering the synergies and conflicts among them [3].

This research in progress aims to analyze whether considering QAs can improve model transformation processes. To that end, we have proposed metrics to measure the *flexibility*, and used them for the model transformation process. It is important to note that the proposed metrics are just an indicator of a QA, and their improvement must not be seen as a complete satisfaction of any QA.

Future work spreads in several directions: (a) proposing more metrics for more QAs considering the complexity of direct and derived metrics: modifiability (*e.g.*, modular designs, dependability of components), analyzability (*e.g.*, data about the errors of the components), performance (*e.g.*, response time), testability (*e.g.*, testing components before applying the model transformation), and consumed resources (*e.g.*, memory); (b) considering constraints in the alternative solutions (*e.g.*, regarding components' technology, location, and provider); (c) assisting software architects to reason about trade-offs on different quality attributes, so that they can prioritize; and, (d) deriving and combining metrics.

Acknowledgments. This work was funded by the Spanish MINECO and the Andalusian Government under TIN2013-41576-R and P10-TIC-6114 projects.

References

1. ACG. ENIA Poject – Development of an intelligent web agent of environmental information, 2011. `http://acg.ual.es/projects/enia/`.
2. Ameller, D., Ayala, C., Cabot, J., Franch X.: Non-functional requirements in architectural decision making. IEEE Software. 30(2), 61–67 (2013)
3. Boehm, B.: Architecture-based quality attribute synergies and conflicts. In: SAM'2015, pp. 29–34. IEEE Press (2015)
4. Criado, J., Rodríguez-Gracia, D., Iribarne, L., Padilla, N.: Toward the adaptation of component-based architectures by model transformation: behind smart user interfaces. Software: Practice and Experience 45(12), 1677–1718 (2015)
5. Herrera, M., Moraga, M.Á., Caballero, I., Calero, C.: Quality in Use Model for Web Portals (QiUWeP). In: ICWE 2010, LNCS 6385, pp, 91–101. Springer (2010)
6. ISO/IEC. ISO/IEC 25000. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE (2014)
7. Loniewsli, G., Borde, E., Blouin, D., Insfran, E.: An Automated Approach for Architectural Model Transformations. In: Information System Development: Improving Enterprise Comm., pp. 295–306. Springer International Publishing (2014)
8. Morin, B., Barais, O., Jézéquel, J.M., Fleurey, F., Solberg, A.: Models@Run.time to Support Dynamic Adaptation. Computer 42(10), 44–51 (2009)