EXTRA-FUNCTIONAL PROPERTIES FRAMEWORK WITH CONFIGURATION BASED ON DEPLOYMENT ENVIRONMENT [*]
Tool Demonstration and Case-study

Kamil Ježek and Premek Brada
Department of Computer Science and Engineering, University of West Bohemia
Univerzitni 8, 30614 Pilsen, Czech Republic

Keywords: Extra-functional properties, Evaluation, Case-study, Tool, Component.

Abstract: Current software systems tend to share not only data but also software parts in a form of software components or web services. Once users cooperate third-party software parts, mechanisms and tools integrating and validating these software parts are desired. Therefore, advanced techniques for verifying compatibility and interchangeability should also consider extra-functional properties. This paper aims at supporting these techniques introducing a tool capable of definition, application and evaluation of extra-functional properties. The ability of the tool is demonstrated in a case-study developed on Common Component Modeling Example.

1 INTRODUCTION

Nowadays, industry and the research community deal with the issue of extra-functional properties (EFPs). For instance, (1) qualities such as *speed*, *response time*, *memory consumption* or (2) user requirements such as *marketability*, *price*, *regular updates*, *technical support* or (3) behaviour such as *synchronisation*, *concurrent access*, *deadlock free computation* are often addressed EFPs.

In our previous work an independent EFP framework (Ježek and Brada, 2011) enabling EFPs in existing industrial systems was proposed. This paper demonstrates that approach in a form of a case-study developed according to Common Component Modeling Example (CoCoME) (Herold et al., 2010).

This paper first introduces the case-study in Section 3 followed by domain EFP definitions in Section 3.1 and context-of-usage dependent EFP definitions in Section 3.2. Finally, Section 3.3 demonstrates the EFP evaluation process.

[*]The work was partially supported by the UWB grant SGS-2010-028 Advanced Computer and Information Systems, by the Grant Agency of the Czech Republic project number 103/11/1489 Methods of development and verification of component-based applications using natural language specifications and by the SPAV grant with the registration number CZ 1.07/2.3.00/09.0050.

2 RELATED WORK

Other approaches target several disjunctive groups of EFPs. One group treats EFPs in isolation (Aagedal, 2001; Franch, 1998; Chung et al., 1999; ISO/IEC, 2001) while another consider their dependency (Zschaler and Meyerhfer, 2003). They provide basic ideas of a structure of EFPs, however, lack of tool support limits their practical usage.

There are structural frameworks with EFPs (Mohammad and Alagar, 2008) or systems with native support of either EFP or quality of service specifications (Yan and Piao, 2009; García et al., 2007) which still focus on a limited set of EFPs.

In addition, there are several component models (Becker et al., 2009; Muskens et al., 2005) which support specialised sets of EFPs. As in our work there are attempts to tool support (Sentilles et al., 2008) or Q-ImPrESS[2] targeted at a selected set of component models.

In our previous work, we stated formal definitions of EFPs (Jezek et al., 2010) which extension and implementation we developed in (Ježek and Brada, 2011).

[2]http://www.q-impress.eu/wordpress/

Ježek K. and Brada P..
EXTRA-FUNCTIONAL PROPERTIES FRAMEWORK WITH CONFIGURATION BASED ON DEPLOYMENT ENVIRONMENT - Tool Demonstration and Case-study.
DOI: 10.5220/0003639403220325
In *Proceedings of the International Conference on Knowledge Management and Information Sharing* (KMIS-2011), pages 322-325
ISBN: 978-989-8425-81-2

3 CASE-STUDY AND TOOL DEMONSTRATION

Case-study is in this paper based on CoCoME that is an example of a trading system with a set of hierarchical components. For the sake of brevity only few components have been selected to demonstrate the tool. The components composing the case-study are shown in Figure 1 with Notes to show attached EFPs.

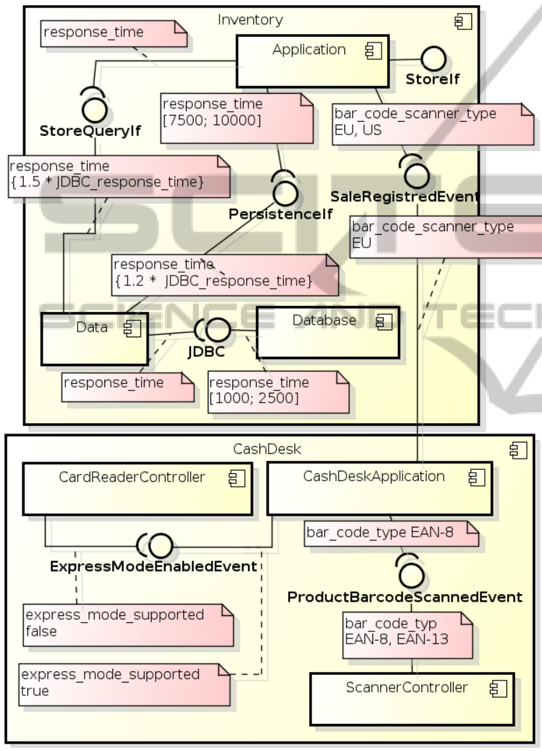


Figure 1: CoCoME Sub-part.

3.1 Domain Dependent Definitions of Properties

The presented tool uses commonly accessible EFP repository spiting domain and context-of-usage concerns. For that, Global Registries (GRs) (Jezek et al., 2010) hold domain specific EFP definitions filled by domain experts.

Figure 2 shows a GUI editor of GR. The tool accesses a remote web-server storing the GR structure (Ježek and Brada, 2011). A user selects GR which he or she tends to work with, then EFPs are edited in terms of their names, data types, comparing functions and named values (in detail formalised in (Jezek et al., 2010)).

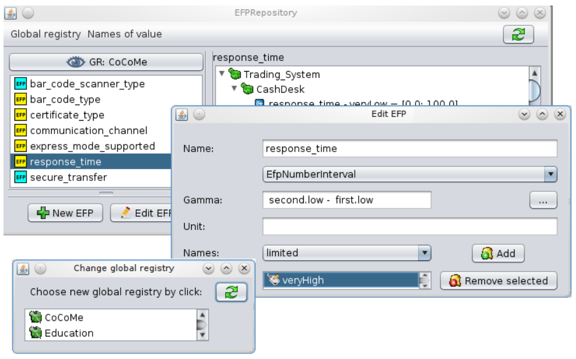

EFPRepository
Global registry Names of value
GR: CoCoMe
bar_code_scanner_type
bar_code_type
certificate_type
communication_channel
express_mode_supported
response_time
secure_transfer
New EFP Edit EFP
response_time
Trading_System
CashDesk
Edit EFP
Name: response_time
EfpNumberInterval
Gamma: second.low - first.low
Unit:
Names: limited Add
veryHigh Remove selected
Change global registry
Choose new global registry by click:
CoCoMe
Education

Figure 2: EFP Registry Tool.

For our study, a GR named *CoCoME* with several EFPs (*response time*, *express mode supported*, *bar code type*, *bar code scanner type*) has been created. The EFPs defined in one GR bounds EFPs as valid and meaningful in one domain. Hence, the EFPs created for CoCoME cover the domain of trading systems.

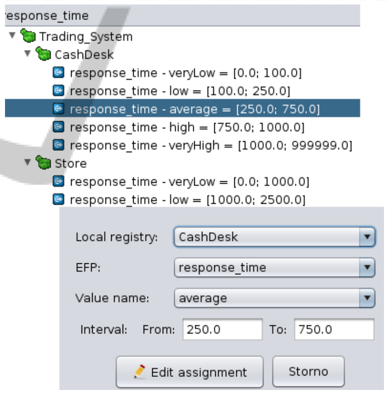

esponse_time
Trading_System
CashDesk
response_time - veryLow = [0.0; 100.0]
response_time - low = [100.0; 250.0]
response_time - average = [250.0; 750.0]
response_time - high = [750.0; 1000.0]
response_time - veryHigh = [1000.0; 999999.0]
Store
response_time - veryLow = [0.0; 1000.0]
response_time - low = [1000.0; 2500.0]
Local registry: CashDesk
EFP: response_time
Value name: average
Interval: From: 250.0 To: 750.0
Edit assignment Storno

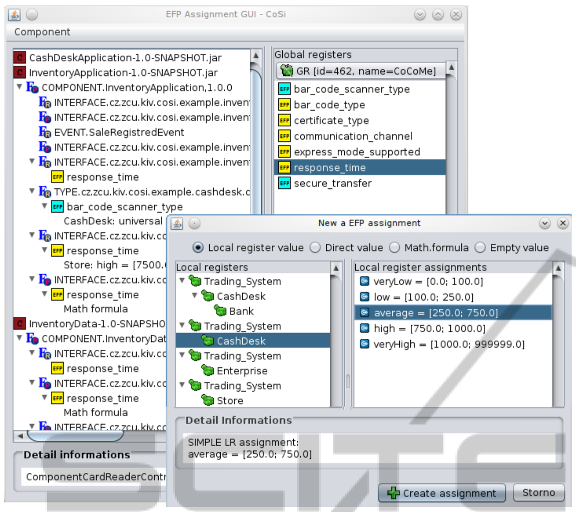
Figure 3: EFP Local Registry Editor.

The context-of-usage concern is captured in a set of Local Registries (LRs) (Jezek et al., 2010) with an editor shown in Figure 3. In the tool a user creates named intervals of values that has two advantages: (1) the "plain" values are given the semantics and (2) values from one interval are unified to the same scale. For instance, *response time* in an interval $\{1000, 3000\}ms$ may be "average" for back-end office (*Inventory* in CoCoME) for all values 1000*ms*, 2000*ms*, 3000*ms* as long as they belong to one group.

3.2 Application of Property Values

Once EFPs are defined in Registries, the tool shown in Figure 4 is used for attaching selected values to components. Components are shown in one panel and

EFPs in the other one. A user drag-and-drops to attach EFPs.



```
EFP Assignment GUI - CoSi
Component
CashDeskApplication-1.0-SNAPSHOT.jar
InventoryApplication-1.0-SNAPSHOT.jar
▼ COMPONENT.InventoryApplication.1.0.0
    INTERFACE.cz.zcu.kiv.cosi.example.inven
    INTERFACE.cz.zcu.kiv.cosi.example.inven
    EVENT.SaleRegistredEvent
    INTERFACE.cz.zcu.kiv.cosi.example.inven
  ▼ INTERFACE.cz.zcu.kiv.cosi.example.inven
      response_time
  ▼ TYPE.cz.zcu.kiv.cosi.example.cashdesk.c
    ▼ bar_code_scanner_type
        CashDesk: universal
  ▼ INTERFACE.cz.zcu.kiv.cc
    ▼ response_time
        Store: high = [7500.
  ▼ INTERFACE.cz.zcu.kiv.cc
    ▼ response_time
        Math formula
InventoryData-1.0-SNAPSHO
▼ COMPONENT.InventoryDat
  ▼ INTERFACE.cz.zcu.kiv.cc
      response_time
  ▼ INTERFACE.cz.zcu.kiv.cc
    ▼ response_time
        Math formula
  ▼ INTERFACE.cz.zcu.kiv.cc
Detail informations
ComponentCardReaderContr

Global registers
GR [id=462, name=CoCoMe]
bar_code_scanner_type
bar_code_type
certificate_type
communication_channel
express_mode_supported
response_time
secure_transfer

New a EFP assignment
(●) Local register value  ( ) Direct value  ( ) Math.formula  ( ) Empty value
Local registers                 Local register assignments
▼ Trading_System                veryLow = [0.0; 100.0]
  ▼ CashDesk                    low = [100.0; 250.0]
      Bank                      average = [250.0; 750.0]
▼ Trading_System                high = [750.0; 1000.0]
    CashDesk                    veryHigh = [1000.0; 999999.0]
▼ Trading_System
    Enterprise
▼ Trading_System
    Store
Detail Informations
SIMPLE LR assignment:
average = [250.0; 750.0]
Create assignment   Storno
```

Figure 4: EFP Assignment Tool.

The tool allows to assign three type of values (Ježek and Brada, 2011):

Context Independent Values. A lot of EFPs express values unchanged among all contexts-of-usage. Here, the tool assigns a concrete EFP value (e.g. number, string, boolean). An example of a property is the *express mode supported* attached to *ExpressModeEnabledEvent*. It is a boolean indicating the express mode of a cash desk which essentially has the same meaning all the time.

Context Dependent Values. Often values vary among contexts-of-usage. Here, several LR named values may be assigned. In the evaluation a user selects the LR to evaluator for. Such a property is *response time* in the case-study. E.g. a cash desk may have harder performance needs than *Inventory*. Hence, the property values are defined in two separated LRs to distinguish two environments.

Computed Values. Complex properties are typically influenced each other (e.g. performance of one component is influenced by a connected components). Here, the tool adds mathematical or logical formulas computing an EFP value depending on other EFPs.

In CoCoME, the component *Inventory::Database*, *Inventory::Data* and *Inventory::Application* are used. Modeling EFPs, the property *response time* on the output of *Inventory::Data* is influenced by the output on *Inventory::Database*. A user may define e.g. a mathematical formula *Inventory::Data*: $response\_time(PersistenceIf) = 1.2 * response\_time(JDBC)$

Another example uses the *bar code type* with values {EAN-8, EAN-13, UPC-A, UPC-E, ...} for existing bar codes. *Bar code scanner type* may be computed by a logical formula: $bar\_code\_scanner\_type = EU \Leftrightarrow bar\_code\_type = \{EAN8, EAN13\}$ where EU is the item of an enumeration.

3.3 Evaluation Process

When components are being assembled into an application, the EFPs of the provided and required features need to be evaluated for the compatibility. This is performed by an evaluator tool which, unlike other tools, does not have a GUI. It is rather implemented as a framework embeddable to other systems. It uses two steps to evaluate EFPs: (1) it binds components via their features and creates a graph expressing their connections, (2) the properties on bound features are compared by the algorithm from (Ježek and Brada, 2011).

For context independent values, the evaluator compares values directly attached on both connected properties. According to the case-study, if the *CashDesk::CashDeskApplication* requires the property *express mode enabled* set to *true*, then a compatible *CashDesk::CardReaderController* component has to provide *express mode enabled* also set to *true*.

Evaluation of context dependent values compares only values from a selected LR where a LR setting is a matter of configuration. Values irrelevant for other LRs are ignored. For instance, values on the properties *response time* for *Inventory* are taken only from *Store* LR while others are ignored.

Computed values evaluation uses depth-first search algorithm to recursively compute properties in a formula. In the use-case, evaluation on *Inventory::Application:PersistenceIf* needs to find *Inventory::Database::JDBC* first. Then a value of *response time* on *Inventory::Data:PersistenceIf* depending on *Inventory::Data:JDBC* may be computed. At the end, the values are compared as for direct values.

4 CONCLUSIONS

In this work a tool concerning extra-functional properties in terms of their definition, application and evaluation has been presented. The tool is based on a framework, developed in our previous work.

The tool aims at improving adoption of extra-functional properties in practice. This aim has been demonstrated by the presented case-study which comprises a part of the Common Component Modeling Example proving the framework's applicability to a practical application.

The case-study has shown that the system of registries is capable of managing extra-functional properties on systems deployed in different runtime environments. On the other hand a few weakness have been found. Namely, there are properties which need more sophisticated evaluating methods (for instance, memory consumption is not a simple summation of each component if they use shared libraries). Such evaluating methods we will target in the future. In addition, filling of values in LRs is a time consuming manual work. For that reason formulas transferring values among LRs would improve the approach.

ACKNOWLEDGEMENTS

We would like to thank students Martin Štulc, Lukáš Vlček and Jan Šváb for their considerable help in preparing the presented case-study.

REFERENCES

Aagedal, J. Ø. (2001). *Quality of Service Support in Development of Distributed Systems*. PhD thesis, University of Oslo.

Becker, S., Koziolek, H., and Reussner, R. (2009). The palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3 – 22. Special Issue: Software Performance - Modeling and Analysis.

Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. (1999). *Non-Functional Requirements in Software Engineering*. Series: International Series in Software Engineering, Vol. 5, Springer, 476 p, ISBN: 978-0-7923-8666-7.

Franch, X. (1998). Systematic formulation of non-functional characteristics of software. In *Proceedings of International Conference on Requirements Engineering (ICRE)*, pages 174–181.

García, J. M., Ruiz, D., Ruiz-Cortés, A., Martín-Díaz, O., and Resinas, M. (2007). An hybrid, qos-aware discovery of semantic web services using constraint programming. In *ICSOC '07: Proceedings of the 5th international conference on Service-Oriented Computing, Springer-Verlag Berlin, Heidelberg 2007, ISBN: 978-3-540-74973-8*, pages 69–80, Berlin, Heidelberg. Springer-Verlag.

Herold, S., Klus, H., Welsch, Y., Rausch, A., Reussner, R., Krogmann, K., Koziolek, H., Mirandola, R., Benjamin, Hummel, Meisinger, M., and Pfaller, C. (2010). Common component modelling example (CoCoME). Book Chapter. Available at http://agrausch.informatik.uni-kl.de/CoCoME/downloads (2010).

ISO/IEC (2001). ISO/IEC 9126: Informational technology - product quality - part1: Quality model, international standard iso/iec 9126, international standard organization.

Ježek, K. and Brada, P. (2011). Correct matching of components with extra-functional properties - a framework applicable to a variety of component models. In *Evaluation of Novel Approaches to Software Engineering (ENASE 2011)*. [accepted to publication].

Jezek, K., Brada, P., and Stepan, P. (2010). Towards context independent extra-functional properties descriptor for components. In *Proceedings of the 7th International Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA 2010), Electronic Notes in Theoretical Computer Science (ENTCS) Volume 264, page 55-71, ISSN: 1571-0661*, pages 55–71.

Mohammad, M. and Alagar, V. S. (2008). TADL - an architecture description language for trustworthy component-based systems. In *ECSA '08: Proceedings of the 2nd European conference on Software Architecture*, pages 290–297. Springer.

Muskens, J., Chaudron, M. R., and Lukkien, J. J. (2005). *Component-Based Software Development for Embedded Systems*, chapter A Component Framework for Consumer Electronics Middleware, pages 164–184. Springer Verlag.

Sentilles, S., Håkansson, J., Pettersson, P., and Crnkovic, I. (2008). Save-ide – an integrated development environment for building predictable component-based embedded systems. In *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008)*.

Yan, J. and Piao, J. (2009). Towards qos-based web services discovery. In *Service-Oriented Computing ICSOC 2008 Workshops, Lecture Notes in Computer Science, 2009, Volume 5472/2009, 200-210, ISBN: 978-3-642-01246-4*.

Zschaler, S. and Meyerhfer, M. (2003). Explicit modelling of qos-dependencies. *In Proceedings of the 1st International Workshop on Quality of Service in Component-Based Software Engineering, Cpadusditions, Toulouse, France*, pages 57–66.