Extra-Functional Properties for Reusable Software Components

Kamil Ježek

17. December 2008
Goal

A component-based development

- a final system composed from components
- provides high-level of reusability
- saves human effort, reduces price, time to deliver, ...
- has not been fully adopted yet

Issues

- extra-functional properties are poorly supported
- a lack of generally accepted formalisms
- a general framework still does not exists

The Main Goal

A general extra-functional properties framework
The Framework for Extra-Functional Properties (EFPs)

- General — from embedded to general systems
- Level of granularity — EFPs bound to whole components or operations only
- Easily usable for programmers
- High level of scalability
- Computer analysable
Extra-Functional Properties Definition

- directly measurable
- directly definable

Basic Property

\[ e_{def}^{basic} = (N, T, M) \] (1)

- **N**: name
- **T**: \( \in \{boolean, integer, float, enumeration\} \)
- **M**: \( string \) — is a measurable unit
Derived Property

Extra-Functional Properties Definition

- derived from basic ones
- defined using simple expression language.
- initially derived only from simple ones, later recursively

\[ e_{\text{def}}^{\text{derived}} = (D, N, \text{enum}); \] (2)

\[ D = \{P_1, \ldots, P_K\} \]

\text{enum} \quad \text{a set of rules expressing a derivation of this property from those simple}
 Derived Property

A Derived Property - The Example

Example

\[ \text{responsiveness}_{\text{enum}} = \{ \text{slow, normal, fast} \}, \]
\[ \text{slow} \iff \text{response\_time} > 1000 \land \text{complexity} > \text{quadratic}; \]
\[ \text{normal} \iff \text{response\_time} < 1000 \land \text{response\_time} > 50 \]
\[ \land \text{complexity} < \text{exponential}; \]
\[ \text{fast} \iff \text{response\_time} < 50 \land \text{complexity} < \text{quadratic}. \]
Registry

- defines the meaning of EFPs.
- stores all meta information related to EFPs

$$R = (loc, \{\text{meta}\}, \{e_i\})$$  \hspace{1cm} (3)

- **loc** is a location URI. It refers to a target namespace
- **meta** = \((name, value)\) is a descriptive field
- \(\{e_i\}\) is a set of (basic or derived) extra-functional property definitions
A Registry - The Example

memory_min : Integer <0, 100> ; "MB"
memory_max : Integer <1, 2000> ; "MB"
complexity : Enum {const, linear, log, power}
available_offline : Boolean;
Name spaces

- define a space in which EFPs are valid
- prevents name clashes
- allows name overloading
- mapping between different variables with the same meaning
Property Descriptor

- contains required and provided properties
- bounds values of EFPs for the concrete component or to operations
- EFPs come from the registry related to the name space
- provided properties are compared only by the equal (\(=\)) operator
- derived properties use comparison operators
  \(\{<, \leq, >, \geq, =, \neq, \subset, \subseteq\}\)
# the association of EFPs to the component

ExtraFunc-Catalog:
   http://services.kiv.zcu.cz/cosi/extrafunc/v1/

Bundle-ExtraFunc:  memory_min = 4,
   memory_max = 50, complexity = log,
   available_offline = false;
- The registry and name spaces mechanism implemented in CoSi [3, 2]
- A mapping between EFPs has not been resolved yet
- A recursive definition of derived properties from derived ones is not currently allowed
- Our approach has been submitted to FESCA 2009
Related Works and Formalisms

- TADL [5]
- A Component Quality Model [1]
- NoFun [4]
- Enterprise Java Bean [6]
<table>
<thead>
<tr>
<th>Goal</th>
<th>Framework</th>
<th>Properties</th>
<th>Registry</th>
<th>Name Spaces</th>
<th>Property Descriptor</th>
<th>Status</th>
<th>Related Works and Formalism</th>
</tr>
</thead>
</table>

Thanks you for you attention
A. Alvaro, E. S. de Almeida, and S. L. Meira.  
A software component quality model: A preliminary evaluation.  

P. Brada.  
The CoSi component model.  

P. Brada.  
The CoSi component model: Reviving the black-box nature of components.

X. Franch.  
Systematic formulation of non-functional characteristics of software.  

M. Mohammad and V. S. Alagar.  
Tadl - an architecture description language for trustworthy component-based systems.  

Sun Microsystems.
JSR220 Final Release.