Extra-functional Properties description for Reusable Software Components

Kamil Ježek
Department of Computer Science and Engineering
University of West Bohemia Pilsen,
Czech Republic
kjezek@kiv.zcu.cz

27th October 2009
Extra-functional Properties for Components

There are some specific problems in a field of component-based programming:

- Components run in different computational environment (here called as contexts)
- A property defined in one context has a different scale in another context
- It is hard to define EFPs (and their values) to be independent of a context
- A context may not be known in a component’s creation time
- Different EFPs are independently important in different contexts
- Earlier approaches shows EFPs languages but does not solve context dependency

A possible solution

There need to be a way of encapsulating context specific values
Our Implementation of Extra-functional Properties (EFPs)

- Two types of properties are distinguished: Simple and Derived ones
- Definition of simple ones inspired from CQML (J. Aagedal 2001) - data types, measurable unit
- Definition of derived one similar to NoFun (X. Franch 1998) - simple rules expressions
- Only constructs for static definitions of properties (data types, derivation rules)
- It will provides EFPs checks before component is used
Context dependent values encapsulation

- System of **registry** containing collections of EFPs
- Global and Local registry records
- Global registry defines EFPs - name and domain type
- Local registry holds values for contexts
- Components align values and scales with Local registries

- Global registry is unique (at least for wide area of usage)
- Each context has records in Local registry - with property’s values
- Not every value must be from Local registry - context independent (physical) values may be assigned directly
Connection of EFPs, Registry and Components

Figure: How registry files are related to components
Figure: Global, Local Registry and Properties
Formalism of Extra-functional Properties

Simple Property

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

Derived Property

\[ e_{\text{def}}^{\text{derived}} = (N, E, T) \] (2)

N  the name of a property
T  ∈ \{boolean, number, enumeration, ratio\} - a domain type
O  ∈ \{increasing, decreasing\}  a direction of an impact
M : string — a measuring unit
E  = \{e_1, \cdots e_N\} properties composing this derived ones
Global registry

\[ GR = (\text{loc}, \{e_i\}) \]  \hspace{1cm} (3)

- **loc**: registry’s URI location
- \( \{e_i\} \): a set of (simple or derived) extra-functional properties
Extra-functional Properties in Local Registry

Local registry

\[ LR = (\text{loc}, \text{loc}_{gr}, S \cup D) \]  \hspace{1cm} (4)

- \text{loc} the URI of the registry
- \text{loc}_{gr} a link to the global registry
- \( S = \{(\text{name}, \text{name}_value, \text{value})\} \) set defining abstract names of values for simple properties from GR
- \( D = \{d_i : \{r_{i1}, \cdots, r_{iK}\}\} \) set of rules expressing that a derived property \( d_i \) is given by \( r_{iK} \) rules
- \( r_i : F \Rightarrow \text{value} \in T; F \) is a logical expression
Example (I)

Example - definition of extra-functional properties

data_transferred : increasing integer KB

time_to_process : decreasing integer ms

data_correct : boolean

# one derived property

performance : data_transferred, time_to_process

    enum {sufficient, insufficient}
Example (II)

Example - definition of values particular for a context

```
time_to_process : high = (500; +INFINITY)
time_to_process : average = (100; 500]
time_to_process : low = (0; 100]

# data_transferred omitted here

performance : sufficient =
  data_transferred >= high AND time_to_process <= low
performance : insufficient =
  data_transferred <= low OR time_to_process > high
```
Attaching Extra-functional Properties To Components

- Each component links to the particular registry
- Only EFPs selected from global registry are allowed

Figure: Assignment of property to component or feature
Example

# the association of EFPs to the component
ExtraFunc-Catalog:
    http://services.kiv.zcu.cz/cosi/extrafunc/v1/

Bundle-ExtraFunc:
    performance = sufficient,
    data_correct = true

Provided-service : cz.zcu.kiv.services.DataReader;
extrafunc=(data_transferred = average,
    time_to_process = low)
Open Issues

- Semantic of value names should be better developed (using ontology, or simply a sequence of names)
- Location of Registry must be determined
- Is it not clear whether only one Global registry can exist
- Composition rules for derived properties has to be improved - they are currently computed at the Local registry level
- A location of registry files need to be considered
- Functions transforming values between contexts is desired

Thank you for your attention