Travel Report

Project DSLpc

09th September, 2008

Travel Details

Destination

University of Minho,
Braga

Date

2th September - 09th September, 2008

Visitors

Tomaž Kosar, University of Maribor (Slovenia)
Matej Črepinšek, University of Maribor (Slovenia)

Travel Purpose

The main purpose for this visit was to startup the project "Program Comprehension for Domain Specific Languages".

Financial Support / Grant

This visit was supported by: GRICES.

Travel Report synthesis

Aims & Objectives

The specific objectives for this travel were to define precisely:

- project objectives;
- the tasks; and
- the schedule.

Achievements

All the objectives listed above, were achieved:

- in general, we discussed the basic concepts involved (domain specific languages and program comprehension) and we agree with the two main research directions as explained below;
- in particular, we discussed about the experiments on assessing DSL and GPL according to Cognitive Dimensions Framework (CDF), as it is also described below;
- as case-studies the following DSLs were suggested: FDL, Graphviz, LINQ, XAML, JavaFX.

Travel Report details

We start the meeting with the presentation of CoRTA paper on *Program Comprehension for Domain Specific Languages* aiming at clarifying the main project objectives and goals. So, we transcribe in the next subsection the main comments of discussion. We conclude the report with a section on project tasks and future work.

After that we felt that was relevant to classify DSLs. The next subsection is devoted to that purpose.

As the first research direction is supported by cognitive dimensions framework, we include another subsection with a short definition of each dimension.

One of the main outcomes of this working visit was precise the relation between cognitive dimensions and language usability and comprehension. The table synthesizing is included in the last subsection.

Comments

We start discussing the presentation *Program Comprehension for Domain Specific Languages*:

- Program Comprehension vs Program Understanding (goals are different). Usually, program comprehension is related with real applications and maintaining. Program understanding is related with program analysis.

- Slide 5 (Motivation) — To introduce a research project aimed at understanding how the standard approaches for the comprehension of GPL can be adapted for the comprehension of DSLs.

  Standard approaches could be improved to deal DSLs, this is, add new concepts related with DSLs.

  Who are the users of program comprehension of DSLs? The developers of the final users of tools. for languages like SQL, spreadsheets formulas and so on, the final users are the final users.

  The main difference between GPL and DSL is that while in GPL a concept can be expressed in different ways while in DSL it is expressed only in a way.

  We don't believe that all tools could be applied to DSLs. It depends on domain. For example, debuggers are always applied to GPL, but in case of DSLs it depends of the purpose of the language.

It is pacific that visualization is useful for PC.

- Slide 8 (Domain Specific Languages) — Languages tailored to specific application domain that offers to users more appropriate notations and abstractions. DSLs are more expressive and are easier to use than GPLs for the domain in question, with gains in productivity and maintenance costs.

  All of us agree with the definition. Level abstraction corresponds, in this case, with the real world.

  Maintenance costs: if we defend that the maintenance costs for DSLs are lower than for the GPL why build up new tools to maintaining this languages? Because, it is a step further. This is, if we have tools to aid at maintenance so it is more one help. It is a crutch.

- Slide 9: add maintenance to the goals.

- Slide 10 (DSLpc Project) — In this project we have the following objectives: to measure how easier is to understand written in DSLs; to understand if existing program comprehension approaches are applicable to DSLs; and to allow the enhancement of DSL program comprehension by enabling user-centric visualization.

  Maybe another new point to add to the goals is to: propose a set of techniques to improve the program comprehension of DSLs.

  In the second point: to understand which one of the existing PC approaches are applicable to DSLs and how could they be improved.

  We should use the first case as basis of the objective, but at the same time we should study the other 3 points. At final, we should be able to use the results to verify how they allow to measure how easier is to understand programs written in DSLs.

- Slide 12 (DSLpc project - how easy is to understand DSL prgs) — The CDF has been used to assess the usability of visual programming languages, while no such study exists for DSLs. Our purpose is to identity the aspects among the CDF that enhanced in the context of DSL.

  The concept of usability is in the sense of if it is easy to learn the language, to develop a program, and to evolve a program.

It was already study the aspects among the CDF applicable to usability and visualization, but not for maintenance.

Concerning Di (dimension i of CDF) a GPL is good or not; a DSL is good or not.

We need to study which dimensions are relevant or not to study/analyze a DSL.

As it does not exist any formal study on advantages of DSLs, our purpose is to identity the aspects among the CDF that are enhanced in the context of DSLs.

- Slide 18 (DSLpc Project — PC approaches data Extraction) — In data extraction: how to define the information to be extracted for each DSL? Maybe we need to restrict the domain/type of DSLs that we want to cover. Because there are DSLs that don't have any associated grammar.

  If a DSL is not supported by a grammar, we need it to change it to support grammars.

  Maybe a classification of DSLs can be useful to determine what kind of DSLs we will support.

- Slide 30 — Maybe we will need to different traversals to the three; or maybe more than one traversal.

  Maybe we need to allow that evaluation rules are also parameterized.

DSL classification

We can classify them according to its level and purpose:

- Specification Languages (the purpose is to model data or operations).

  Examples: JML, UML, BML, LISA specification language (LISAsl), FDL, make, yacc, lex, AWK, JavaFX, DTD, Dot, XSD, SGML, XML ...

- Programming Languages (the purpose is to produce some actions/results).

  Examples: SQL, LINQ, XSLT, XPath, XQuery, PIC...

  We call DSPL to Domain Specific Programming Language, a subset of DSL, and this subset can be compared and related with GPL (General Purpose Language) with API (Application Program Interface).

- Annotation languages (complementary descriptions).

  Examples: HTML, XAML, SVG, GML, ASP, JSP ...

Cognitive Dimensions (Definition)

Abstraction Gradient

What are the minimum and maximum levels of abstraction?

In a DSL, all operations have a zero gradient because all operations are at the same level. In a programming language like the C language it have a high gradient because we can work at bit level.

Consistency

When some of the language has been learnt, how much of the rest can be inferred?

Error-proneness

Does the design of the notation induce "careless mistakes"?

Hidden-dependencies

How dependencies among program components are immediately visible? How dependencies are easy to detect?

Imposed guess-ahead

Do programmers have to make decisions before they have the information they need?

Progressive evaluation

Can a partially-complete program be executed to obtain feedback on "How am I doing"?

Role-expressiveness

How each component of a program is related to the whole?

Viscosity

How much effort is needed to perform more changes?

Visibility/Juxtaposibility

Is every part of the code simultaneously visible or it is at least possible to compare any two parts side-by-side at will?
If the code is dispersed, is it at least possible to know in what order to read it?

Closeness of mapping

How closely does the notation correspond to the problem world?

Diffuseness

How many symbols or how much space does the notation require to produce a certain result or express a meaning?

Hard mental operations

How much hard mental processing lies at the notation level, rather than at the semantic level? Are there places where the user needs to resort to fingers or pencilled annotation to keep track of what's happening?

Secondary notation and escape from formalism

Can the notation carry extra information by means not related to syntax, such as layout, color, or other cues?

Cognitive Dimensions vs Language Usability and Comprehension

First the guidelines where classified according with its influence type: positive or negative. From the initial 13 guidelines we believe that not all of them will be tested by experiment.
After that, it was decided to study each one of this guidelines in 4 fields: how they contribute to learn, to comprehend, to evolve and to develop.

In a scale from 0 to 5 we evaluate each of one of this guidelines. Zero means that have no influence and 5 means direct influence.

The final result is listed in table 1.

Dimension	Influence	Learn	Comprehend	Evolve	Develop
Closeness of mapping	+	3	3	3	3
Viscosity	-	0	0	5	5
Hidden dependencies	-	1	1	3	3
Hard mental operations	-	3	3	2	2
Imposed guess-ahead	-	3	3	4	4
Secondary notation	+	3	3	3	3
Visibility	+	3	3	3	3
Consistency	+	4	4	4	4
Diffuseness	-	3	3	3	3
Error-proneness	-	0	0	3	3
Progressive evaluation	+	0	0	2	4
Role expressiveness	+	3	3	3	3
Abstraction gradient	+	3	3	3	3

Tabela 1: Nice table! It works. Do you agree Marjan?

We decided not to include the development of DSLs in our study because:

- Influence of development environment (both GPL and DSPL environment can provide comparable tools)
- Our focus is more about comparison of DSPL and GPL learning, understanding and comprehending programs and basic documentation;

We decide to define a minimum set of different questions to evaluate a DSL. So, a DSL study will always have as basis these questions.

Questions:

Learn

- Select correct statements — syntax based question;
- Select statements with no sense (unreasonable) — semantic based question;
- Select valid programs for a given result — meaning based question.

Comprehend

- Select correct results for a given program — valid paths in result set;
- Select number of different results produced by a given program — number of valid paths;
- Select programs with the same result — equivalent classes.

Evolve

- Add a new functionality — expand based question;
- Inhibit a functionality — remove based question;
- Change a functionality — replace based question.

The values in the table 1 should be revised after using the questions above, and also should be tuned for each domain.

Project tasks

The tasks are:

1. to understand and measure how easy is to handle and comprehend programs written in DSLs;

2. to adapt and improve Program Comprehension Tools for DSLs programs.

Outcomes (first task)

1. In which cognitive dimensions DSLs apport an effective gain?

2. Confirmation of the hypothesis that DSLs are easier to use (learn, develop, evolve) and to comprehend?

3. Identification of the dimensions where comprehension adds are specially important.

Outcomes (second task)

1. Confirmation of the hypothesis that classic approaches for GPLs program comprehension can be reused for DSLs;
2. Identification of the approaches and techniques more adequate for DSLs.
3. Improvement of DSLs program comprehension tools.

Global outcomes

The final goal of the project is the identification of tools (the convenient approaches) that can be a good answer to implement the adds that we identify in the first task (item 3).

Future work

Write two papers: one directed to the first research task; and the other one directed to the second research task.