Travel Report

"PCVIA–Program Comprehension by Visual Inspection and Animation" Project

14th December, 2007

Travel Details

Destination

University of Minho, Braga–Portugal

Date

12 to 14 December, 2007

Visitors

Árpád Beszédes

Travel Purpose

The main purpose of this visit was to discuss and assess the R&D project PCVIA.

Financial Support / Grant

FCT under contract No. POSC/EIA/57662/2004.

Travel Report

Aims & Objectives

The objectives were:

1. to introduce both research groups and their main areas of interests
2. to discuss tasks involved in PCVIA:
   - Alma architecture and implementation
   - PICS architecture and implementation
   - WebAppViewer architecture and implementation
3. to have the consultant's comments on PCVIA project
4. to identify common topics for future cooperation

Achievements

The reported working visit was very successful, as the above objectives were completely attained:

- Concerning the first item:
  - Pedro Henriques gave a talk about University of Minho, Informatics Department and the language processing research group.
  - Maria João Varanda presented PCVIA project, the tasks involved, the work undergoing and the results achieved.
  - Árpád Beszédes gave an overview about the University of Szeged and his work at the Department of Software Engineering.
- Concerning the second item:
  - Daniela da Cruz talked about Alma architecture and showed full implementation of the PCVIA strategy on an example of visualizing a Domain Specific Language by Abstract Syntax Trees and their operational semantics.
  - Mario Berón has already come back to Argentina, so Pedro explained his work and Maria João presented PICS system. PICS includes debugging and comprehension features and it follows a code instrumentation approach.

  - Ruben presented a demo of WebAppViewer and Eva Oliveira emphasized the requirements she defined for this tool in her master thesis, in order to give a real help on comprehension of web applications.

- Consultor comments (mentioned above in the third item) are:
  PCVIA was a research project, whose main objective was to design new methods for program comprehension, in which two main instruments are to be used for the comprehension: visualization and animation. The project's main goals were to study state-of-the-art in program comprehension, to design new methods, implement them in prototype tools and finally to experimentally assess the usefulness of the approaches. The primary outcome of the research is documented in scientific publications at conferences and in journals of the field.
  Overall, the objectives of the project have been achieved. Namely, different methods and tools have been elaborated which address program comprehension from different perspectives. The survey performed in the beginning of the project should be published in an approapriate publication. Specifically, three tools are described:

  - ALMA, which is a general, flexible framework for visualization and animation of program concepts based on a unified internal representation and mechanisms for visualization and animation, all using attribute grammars. A specific application of the framework is implemented for animating the simulated execution of procedural programs/algorithms.
  - PICS, a program understanding framework for C programs, which uses different levels of abstraction for presenting program artifacts, including static and dynamic views.
  - WebAppViewer, a comprehensive toolset for comprehending existing web applications, currently for PHP technology. The tool includes different views on the program, both static and dynamic.

  The common property of the approaches is that they aim to separate the problem domain from the program domain, that is concepts from operation. This is very important since the basic objective of program comprehension is to create a mental model of the system investigated, and this can only be achieved by separating operational program objects from problem domain concepts encoded in the system. Any (semi-)automatic method that can help with this mental process is

welcome.
The above-mentioned outcomes of the project clearly provide a proof-of-concept, and they establish a solid ground for further research. In a way, a good infrastructure has been developed, but for fully exploiting the results further research is needed. It is to be shown how these concepts can be aplied in different, more realistic situations. For example, the ALMA approach could be used in different contexts other than animating algorithm execution and WebAppViewer should be verified in real projects.
Detailed suggestions for the improvement of the methods:

- ALMA - The framework provides a good theoretically-founded basis based on attribute grammars, and built-in libraries, which can be used for different purposes in which custom-designed visualization and animation of program entities are applied. Currently, only low-level programming concepts related to procedural languages are implemented with the capability of animating their execution. This resembles program debugging, however there are several significant differences. Most notably, with ALMA, it is possible to investigate program 'execution' using abstract concepts and not only program-level instructions.
  However, the current implementation lacks several important features to be applicable for more realistic applications, these include: handling of pointers, unstructured control flow, runtime libraries, I/O operations, etc.
  Another suggestion for improvement would be to try integrating internal representations that are more close to problem concepts (like an abstract semantic graph), rather than using abstract syntax tree that follows the grammar used for parsing the program. It often inherently contains syntactic solutions required by the language processing method used, but which represent high level concepts in a difficult way. It could also be considered to use an existing general program representation, like GENERIC which is used by the GCC compiler, or the Columbus schema.
  An interesting enhancement of the current execution animation of ALMA would be to integrate dynamic slicing, in which the slices could represent different concepts or tasks (as they describe different computations, basically). Furthermore, an interesting side-result of this would be the definition of slicing algorithms with the help of attribute grammars.

The flexible architecture allows different applications in which custom visualization elements and animation purposes can be defined. This way, both the problem and the program domain can be appropriately represented. So, it would be straightforward to verify the approach in other applications. For example, animating higher level models (like class diagrams) in the context of change propagation or test coverage measurement, just to name few.

– PICS - PICS uniquely provides the possibility to investigate the subject system given with its source code in both directions of abstration. Namely, looking at the lower-level assembly code, and higher level relationships identified from the code like call-graphs. This concept is extended with several interesting views that can help the comprehension in various ways. However, the tool in it current state should be further enhanced to be more applicable in real life situations. Probably the BORS part needs the most rethinking since in its current form it gives a very simplistic view on 'understanding' a function by its dynamic callees.
Future goals with the approach are not so clear, here are several further enhancement possibilities in the tool: - the way multiple execution of sequences due to loops can be handled using the compression of the trace rather than simply limiting the number of iterations. Examples of trace compression are the Sequitur algorithm and Binary Decision Diagrams. - a possible way to approximate behavioral concepts is to measure the dynamic function coupling from the traces: http://www.inf.u-szeged.hu/ beszedes/research/BGF07.bib - also for the BORS approach, it seems that information from program specification cannot be avoided in some form or another. - different kinds of coverage measurement on function level could be used with the dynamic call tree and the static call graph of the system. Namely, node-, branch- and path-coverage could be measures for the executions.
– WebAppViewer. The tool builds on various existing components, hence it provides rich features combined in a very intuitive, comprehensive tool. Although the benefits compared to other similar tools are not so clear, even this prototype version can be a very useful aid for understanding the working of certain types of web applications. The integrated execution of pages and the possibility for the investigation of dynamic views is very plausible.

Further enhancement of the tool includes the possibility to be used even in the software development phases, in which case rapid analysis of modified code should be solved. Furthermore, by the summarization of different executions upon different web requests could provide a means to measure and visually analyze the achieved coverage in terms of structural code elements.

- Future cooperation topics are:
  - To exchange the tools developed in both groups (http://www.frontendart.com/products.php);
  - Exploring the idea of use a meta-language for internal representation schema to generate implementation source code;
  - To apply slicing to grammars, based on the idea of 'rule dependency graph', in a way that we can answer questions like: "if I modify this rule what other rules will be affected?";
  - Implementation of slicing algorithms using attribute grammars;
  - To use ALMA for: high level (UML class diagram, language independent representation), animation for testing or change propagation; implementation of Hungarian dynamic slicing algorithm (http://www.inf.u-szeged.hu/ beszedes/research/BGG06.bib);
  - To investigate aspect-oriented specifications versus slicing;
  - To enhance PICS, using of Columbus schema (experience in instrumenting C code) (http://www.inf.u-szeged.hu/ beszedes/research/BGS01.bib), DFC, coverage measurement;
  - To use the experience of Hungarian group in metrics, coding rules, bad smells and testing for improve our tools.