OASIS: OPENING-UP ARCHITECTURES OF SOFTWARE-INTENSIVE SYSTEMS

M. Lizotte*
Defence R&D Canada – Valcartier
Val-Bélair, Qc, Canada, G3J 1X5

J. Rilling
Concordia University,
Montreal, Qc, Canada, H3G1M8

ABSTRACT

Opening-up architectures of software-intensive systems includes, as a key element, reverse engineering software, up to a simple component level. This paper introduces the Oasis project, which aims at decreasing comprehension time of existing systems to be used in a system-of-systems. The problem to be addressed and the vision are presented and current tool deficiencies are described.

1. INTRODUCTION

Understanding architectures of existing system-components is a long and complex task slowing down development of system-of-systems (SoS). The current project aims at developing advanced methodologies, techniques and tools to accelerate such comprehension of existing systems required to be integrated as components of a SoS. A SoS can be defined as a system built from components that are themselves large systems. Each component system serves a purpose on its own before, during, and after, being integrated into the SoS. In addition, these component systems are managed by different organizations, i.e., they are acquired and they evolve independently. The specific project objectives are:

- To perform a state of the art of technologies enabling such comprehension and outline current deficiencies;
- To describe an innovative concept for speeding up comprehension;
- To develop a proof-of concept prototype; and
- To experiment the value of this concept.

2. BACKGROUND

For the past several years, system and software architecture has been a focal point of many research efforts. However, most of this ongoing research is focusing on the new development (forward engineering) of software systems, rather than addressing issues in maintaining, analyzing and comprehending existing system architectures.

Software maintenance of existing systems consumes 50-70% of the total programming effort and a significant portion of this maintenance activity (30-60%) is spent on software understanding (Lientz and Swanson, 1980). The increasing complexity of software systems is aggravating maintenance of these systems.

Architectural recovery can be seen as a discipline within the reverse engineering domain that is aimed at recovering the structure and relationships of a system's components. It goes further than design recovery with its specialized focus on the recovery of lower-level aspects, e.g., class design. Performing the architectural recovery process manually is often too time consuming and error prone. Architectural recovery tools need to be applied to support the recovery process, although human intervention and interpretation of tool-extracted artifacts are important issues. This is because architectural mechanisms are seldom found directly in artifacts; rather, they exist in abstractions and compositions of extracted information.

3. PROBLEM DESCRIPTION

Understanding existing systems is one of the main challenges facing a System of Systems (SoS) architect. It is inherent in the complexity of assembling heterogeneous systems. Architects who master these system components are rarely available. Architectural models are often obsolete, inaccessible or nonexistent. Documentation is sometimes up-to-date, but frequently voluminous and much too detailed to satisfy specific needs of the architect, such as specific abstraction level and viewpoint. The current project addresses this comprehension issue within the context of SoS development where systems being integrated are in constant evolution.

4. VISION

The vision can be expressed with an analogy: a SoS architect is similar to a doctor having different tests and examination techniques available to understand the

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
00 DEC 2004	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
OASIS: Opening-Up Architectures Of Software-Intensive Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Defence R&D Canada Valcartier Val-Bélair, Qc, Canada, G3J 1X5; Concordia University, Montreal, Qc, Canada, H3G1M8**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001736, Proceedings for the Army Science Conference (24th) Held on 29 November - 2 December 2005 in Orlando, Florida.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**2**	

patient problem. The elaboration of a solution is enabled from the results of some of these tools. This vision can be summarised as follow:

- The system of systems architecture team has a toolbox to analyse an existing system.
- Architects select tools to be used according to the needs such as abstraction levels and viewpoints.
- Architects get an up-to-date, reliable and appropriate representation of the system architecture.
- The architecture team can elaborate a sound SoS solution faster using relevant information provided by the tools.

5. CURRENT DEFICIENCIES

Based on the complexity and variety of existing architectures, even with the improving maturity of current reverse and re-architecting tools, most tools lack support for essential features. Parsing techniques and limited static analysis are becoming more widely available in current tools. However, there still exists a major shortcoming in the ability to utilize dynamic tracing and analysis techniques to support the architectural recovery process of today's distributed and dynamic systems. The support of dynamic analysis and visual representation of dynamic information is either completely lacking or only partially supported by current tools. Clustering, grouping algorithms and feature extraction have to be enhanced to provide additional meaningful and required views of the system under investigation. Furthermore, there exists the need to extend architectural recovery techniques, tools and processes to incorporate domain and user knowledge to provide support for the partial reconstruction of operational and technical aspects.

Most existing tools are based on traditional reverse engineering approaches relying on some type of source code parsing. Information extraction techniques depend on the type of information that needs to be extracted. Several tools support domain knowledge and user interactions (e.g. O'Brien, 2002 and Finnigan et al., 1997)) to extend the knowledge base with non-source specific information essential for a successful architectural recovery. Most existing tools store extracted information in some type of repository, usually based on a SQL database to simplify processing of the data views. The software visualization techniques used by most recovery tools are another carry over from the more traditional reverse engineering tool domain. Most of the tools provide support for UML visualization based techniques while some support still more procedural orientated visuals, like call-graphs, tree structures. Analytical support is essential in organizing; accessing and analyzing source code information and implementation dependencies. The majority of the tools lack detailed analytical support. Their analysis support is limited to standard high-level dependencies and metrics and mostly to static analysis with some exceptions (e.g. (Sartipi and Kontogiannis, 2001). The complexity and variations that exist in software systems make it impossible for one tool to provide a universal solution for all problem domains and comprehension tasks. One of the ongoing trends in building architectural recovery tools is towards plug-ins and extensible interfaces, often leading to open tool frameworks and workbenches (e.g. O'Brien, 2002 and, Finnigan et al., 1997). These tools normally support standard knowledge bases, common data exchange formats and well-defined interfaces for external plug-ins as well as scripting support to make these tools highly customizable and extensible.

6. CONCLUSION

The Oasis project started in 2003. In collaboration with Concordia University, a State of the Art was conducted. In addition, DRDC reviewed nineteen tools performing some software reverse-engineering. DRDC studied the impact of metric-based software reorganization on comprehension (the equivalent of software refactoring at the component level). DRDC has also outlined a generic "manual" process followed by software architects to comprehend architectures in the context of a SoS development. The goal of Oasis is to ameliorate this process with integrated toolsets and (in particular) lessons learned regarding toolset limitations and effective utilization guidelines. The next steps are to:

(1) experiment architecture comprehension with a commercial tool;
(2) refine the architecture comprehension process;
(3) develop an approach addressing areas that are the most time-consuming and error-prone for the architect; and
(4) build a prototype to demonstrate the approach.

REFERENCES

Finnigan, P.J., R. Holt, I. Kalas, S.Kerr, K. Kontogiannis, H. Mueller, J. Mylopoulos, S. Perelgut M. Stanley, and K. Wong, 1997: The Portable Bookshelf, *IBM Systems Journal* **36**, 4, 564-93.

Lientz, B.P., and E.B.Swanson, 1980: Software Maintenance Management. Addison-Wesley.

Sartipi, K., and K. Kontogiannis, 2001: A Graph Pattern Matching Approach to Software Architecture Recovery. *Proc., IEEE Int. 86 Conf. on Soft. Maint.* Florence, Italy, IEEE Computer Society. pp. 408-419.

O'Brien, L., 2002: Experiences in Architecture Reconstruction at Nokia [Available online at http://www.sei.cmu.edu/publications/documents/02.reports/02tn004/02tn004.htm