Managing Configuration of Update-enabled Software Components*

Vladimír Mencl

mencl@nenya.ms.mff.cuni.cz
Department of Software Engineering
Faculty of Mathematics and Physics
Charles University in Prague, Czech Republic

Abstract. Current component technologies provide only a primitive support for configuration management. For applications composed of components provided by independent vendors, it should be expected that parts of the application can evolve independently on other parts. Strong requirements are put on the underlying infrastructure, e.g., high-probability/continuous availability of the system, transparency of updates with respect to both the component clients and designers. Furthermore, maintaining a history of updates and providing a support for reverting to a previous configuration after an unsatisfactory update is desired.
For co-existence of component implementations and handling of updates, advanced schemes of the component configuration should exist, considering, e.g., the autonomous nature of components developed by independent vendors, as well as evolution of the component specifications, and transfer/conversion of component state among component versions.
We aim at developing a scheme of component models' extension, which would enrich an existing component model by a support for co-existence of different implementations of a component (either variants or updated versions), transparent updates (utilizing the co-existence of both the new and the old version), and transfer/conversion of the component state. Selection of the components suitable for initiating an update, as well as component specification evolution, will be addressed.

Keywords software components, updating, architecture description languages (ADL), software configuration management

1 Introduction

Currently, in component composition concepts and tools, component systems provide only a marginal support for component evolution. In particular, autonomous evolution is required for cooperating components designed resp. provided by independent vendors. This is true for commercial [6, 7] systems; however

* This work was partially supported by the Grant Agency of the Czech Republic – project 201/99/0244

neither the systems originated in academia [9, 10] do address the assembly process directly, especially with respect to the composition of components provided by independent vendors. In fact, web services can be considered as a special case of such autonomous components. Therefore addressing this issue is of major practical importance.

In the following section we present our goal statement identifying the issues to be addressed. Finally the concluding section reports on the current status of the research.

2 The Problem Addressed

Problem Description (What is the exact problem that this research attempts to address? What are the limitations or failings of current knowledge, methods or technologies that this research will resolve? How significant is this problem?)

Answer (what is a problem): Current technologies provide only a primitive support for managing configuration of applications composed of software components, addressing only the trivial cases (replacing the whole application, replacing only a single library, ...). Moreover, merely marginal support is provided for runtime updates, transition of state and/or evolution of specifications.

Due to this, configuration of complex applications is hard to manage, particularly, e.g., coexistence of multiple versions of a component shared by several applications, handling of concurrent *variants* of implementation, or reverting to a previous version after an unsatisfactory update. Therefore, managing complex configuration takes a high overhead as to (human) resources and increases the overall cost of the software application.

Answer (why the problem is a problem): As components can be deployed in an arbitrary number of software installations and strict requirements can be put on the quality of service provided by the component (e.g., high/continuous availability), proper measures must be taken, such as, e.g., providing runtime updates. These issues are closely related to other fields of updating, e.g., transfer of state, and/or evolution of specifications (possibly with coexistence of components adhering to different specifications). As applications can be composed of components developed by independent vendors; components from different vendors are likely to evolve independent of each other. To consistently handle updates of such components, component composition paradigms must be enhanced and reflected in the component assembly infrastructure.

Goal Statement (What new knowledge, methods, or technologies will this research generate? How will these developments solve the identified problem?)

Answer (how do we address the problem): We aim at developing a scheme of extensions to component models, which would provide advanced facilities in component configuration management. The scheme will consider and handle variants [12] of component implementations, updates to components

adhering to a new version of a specification, as well as selection of components in an application/component hierarchy suitable for initiating an update. The update facilities would also handle runtime updates for components, performing state transition, as well as providing the maximal possible availability of the component (by employing co-existence of the old and the new version where possible and suitable). Component specifications will support these tasks (e.g., behavior specification can be employed [5, 15]). To address the autonomous nature of components provided by independent vendors, different modes of composition and multiple levels of depth of component specifications will be considered; this is reflected in the concept of *autonomous points*.

Approach to be Used (What are the planned experiments, prototypes, or studies that will be done to achieve the stated goal? How will achievement of the goal be demonstrated and the contribution of the work be measured?)

Answer (how do we address it − approach): A proposal of component model extensions will be elaborated in the thesis and published; a prototype implementation will be available.

The scheme will be feasible to implement for a component model satisfying certain basic requirements (ADL-based component descriptions, component composition, possibly behavior specifications); component models to be considered are CCM [7], Darwin [14], Wright [13], or the SOFA component model [1]. As a proof-of-the-concept, a prototype implementation of key tools will be developed for SOFA [1, 4, 5].

3 Current Status & Future Work

Preliminary results have been obtained in analyzing update scenarios with respect to the component life-cycle; these are presented in the *"Autonomous Points in Component Composition"* poster. Previous research was focused on enriching an ADL with support for specification evolution; we addressed the issue with a repository [3]; a prototype implementation of is available.

The work in progress is an assembly tool considering the recent results (presented in the accompanying poster), i.e., supporting different modes of component composition, related to the level of depth of specification of a particular component's interface and postponing the subcomponent selection into a later life-cycle stage as appropriate for the respective component composition mode.

The research has so-far focused on component configuration issues; state transition and dynamic component updating will be covered at a later stage.

References

1. Plášil, F., Bálek, D., Janeček, R.: *SOFA/DCUP: Architecture for Component Trading and Dynamic Updating*, Proceedings of ICCDS'98, May 4-6, 1998, Annapolis, Maryland, USA, IEEE CS Press 1998

2. F. Plášil, D. Bálek, R. Janeček: *DCUP: Dynamic Component Updating in Java/CORBA Environment*, Tech. Report No. 97/10, Dept. of SW Engineering, Charles University, Prague
3. Mencl, V., Hnětynka, P.: *Managing Evolution of Component Specifications using a Federation of Repositories*, Tech. Report No. 2001/2, Dept. of SW Engineering, Charles University, Prague
4. Bálek, D., Plášil, F.: *Software Connectors and Their Role in Component Deployment*, Proceedings of DAIS'01, Krakow 2001, Kluwer 2001 (in print), 18 p.
5. Plášil, F., Višňovský, S., Bešta, M.: *Bounding Component Behavior via Protocols*, In proceedings of TOOLS USA '99, pp. 387-398, Aug 1999.
6. Matena, V., Hapner, M.: *Enterprise JavaBeans 1.1 Specification*, Sun Microsystems Inc., August 10, 1999
7. Object Management Group: *CORBA Component Model Specification*, orbos/99-07-01,
8. Object Management Group: *CORBA 2.4.2 specification*, formal/01-02-33, `ftp://ftp.omg.org/pub/docs/formal/01-02-33.pdf`
9. Levans, G.T., Sitamaran, M.(eds): *Foundations of Component-Based Systems*, Cambridge University Press, 2000, ISBN 0-521-77164-1
10. Medvidovic, N., Rosenblum, D. S., Taylor, R. N. : *A Language and Environment for Architecture-Based Software Development and Evolution*, ICSE-21, Los Angeles, May 1999
11. R. H. Reussner *Enhanced component interfaces to support dynamic adaption and extension*, HICSS-34, IEEE, Jan. 3–5 2001.
12. Conradi, R., Westfechtel, B.: *Configuring Versioned Software Products.*, Ian Sommerville (Ed.): Software Configuration Management. ICSE'96 SCM-6 Workshop, Berlin, Germany, March 1996, LNCS, Springer-Verlag 1996
13. Allen, R., Garlan, G.: *Formalizing Architectural Connection*, In Proceedings of the Sixteenth International Conference on Software Engineering, pages 71-80, Sorrento, Italy, May 1994.
14. Magee, J., Dulay, N., Kramer, J.: *Regis: A Constructive Development Environment for Distributed Programs*, In Distributed Systems Engineering Journal, September 1994
15. Lea, D., Marlowe, J.: *Interface-Based Protocol Specification of Open Systems using PSL*, In proceedings of ECOOP'95, Denmark, Aug 1995, Springer LNCS 952
16. Feiler, Peter H.: *Configuration Management Models in Commercial Environments*, CMU/SEI-91-TR-7. Carnegie Mellon University, Pittsburgh, PA, 1991
17. Larsson, M., Crnkovic, I: *New Challenges for Configuration Management*, Proceedings of SCM-9, Toulouse, France, September 1999. LNCS 1675 Springer-Verlag
18. P. Brada: *Component Change and Version Identification in SOFA*, Proceedings of SOFSEM'99, Nov 27 - Dec 4, 1999, Milovy, Czech Republic, Springer LNCS 1725
19. F. E. Redmond III: *DCOM: Microsoft Distributed Component Object Model,* IDG Books Worldwide, Inc.,Foster City, USA, 1997