Component-Based Application Architecture for Enterprise Information Systems

Erich Ortner

Technische Universität Darmstadt, Germany
ortner@winf.tu-darmstadt.de

Abstract. The paradigm of reuse is a traditional concept of surviving for humanity that manifests itself in human languages. The words (components) will be taken out of a lexicon (repository) and then combined to sentences (applications) according to the rules of a specific syntax (grammar). The paper points out the parallels between the component-based approach of human languages on the one hand and component-based application-system design in the software-engineering discipline on the other hand. We describe some instruments (e.g., repositories, part lists) for managing component-based system design, and introduce a language-critical middleware framework supporting the development and processing of component-oriented e-commerce applications (e.g., an electronic marketplace for trading software components). Furthermore, we present a classification of component types and a component specification framework. The existence of standards and exchange forums (e.g., market places) is — besides a sophisticated component- and configuration theory — a substantial prerequisite for superior component-based application development and system life-cycle management.

1 Introduction

The idea of component-based development of application systems was first introduced into the debate about a construction tenet for database applications in [22]. Before that it was discussed as a possible approach to software engineering from an economic point of view in 1968 by McIlroy [6].

Only when large software vendors such as Microsoft, Sun Microsystems, IBM, SAP, Oracle started to change their products into component-based solutions was component orientation taken seriously in computer and information science. From a theoretical point of view, the term "software component" is not yet considered a technical term in computer and information science, when, for example, compared with the term "machine element" in mechanical engineering.

T. Härder and W. Lehner (Eds.): Data Management (Wedekind Festschrift), LNCS 3551, pp. 181-200, 2005.

The component paradigm in language-based computer and information science [10] can—compared with natural languages used by human beings—in general be described as follows:

> The components (words) are retrieved from a repository (lexicon) and are then put together as applications (sentences) in accordance with a syntax (grammar) to achieve a particular goal, for example, to accomplish a task.

Example 1: Human languages

In order to air a room, one person X says to another person Y:

> "Y, could you please open this window?"

Example 2: Application systems development

For organizing a sales information system, the following (software) components are connected in mutual functional dependence:

> Together, a program for maintaining customer data, one for maintaining orders, and a billing program combined with accounts receivable, accomplish the tasks of a sales information system within an enterprise.

Based on a grammar you can create almost any number of new sentences by reusing words (components), which can thus express new associations of ideas (ways of carrying out tasks). Linguists and paleoanthropologists even think that along with the continued evolution of grammar in human languages (so called "natural languages"), creative power and imagination (creativity) in human beings developed accordingly. The linguist Derek Bickerton expressed this view in an interview [1] as follows: "If, for example, you look at barbs, be it barbs for harpoons or else for darts...—Believe me: there is a great deal of grammar in the development of the barb". And the physicist Albert Einstein (1879-1955) when asked about thinking and language is believed to have said: "A thought occurs to me, and then I have difficulties in expressing it."

Fulfilling the component paradigm in application system development requires a variety of unambiguous definitions. We need to systematically build component catalogs, clearly define the functional scope of (software) components and reconstruct a "rational grammar" [5] in order to connect components with natural language systems, and we need to establish quality standards for components and systems worldwide, to be able to communicate in a digitalized world (interactions via cyberspace). Only when all this has been achieved, we can really speak of a construction tenet for application systems. Looking at it in an integrated way, such a tenet always consists of linguistic components (for example, knowledge and software) and non-linguistic components (physical things and occurrences). It is becoming more and more important (because of ubiquitous computer aid, e.g., [18]), to partly redefine the science of engineering—including its youngest child, computer science—from the point of view of language theory and (constructive) logic [3].

Regardless of the discipline: mechanical engineering, electrical engineering, civil engineering or computer science [23], when construction processes are to be computer-aided, "construction" is always considered a kind of "text modification" (from the specification of requirements to construction plans) based on language theory (e.g., grammar) and constructive logic.