Process of Product Fragments Management in Distributed Development

Darius Strašunskas and Sari Hakkarainen

Dept. of Computer and Information Science, Norwegian Univ. of Science and Technology
Sem Sælands vei 7-9, NO-7491 Trondheim, Norway
{dstrasun; sari}@idi.ntnu.no

Abstract. Management of product constituent fragments is essential for large scale logically or physically distributed projects. Geographically distributed development projects have special settings and needs – special attention has to be given to artifacts management because developers are likely to use different representation formats and a variety of tools for the artifact production. The question is: how can artifacts in different representation formats be related and managed?

Methodological support for artifacts management and traceability is presented in this paper. Product fragments from different development phases (i.e., requirements specification, design, code, test scenarios, and documentation) are interrelated through a conceptual domain model. Domain model is proposed as a means to capture information content despite heterogeneous representation. Given, a domain model with intra-related concepts and artifacts associated to the concepts we are able to interrelate heterogeneous artifacts and to predict and assess how one altered artifact may impact other artifacts. The approach covers the whole lifecycle of a system, enables artifacts’ management by associating them according semantics contained inside.

1 Introduction

Information system development is a highly iterative process, in which developers seek to capture the needs and desires of all stakeholders. The goal is to transform the requirements into a complete system, consisting of both manual and computerized parts. The product of a development project undergoes changes because of its iterative nature. Management of the development process imposes requires fine-grained control over all fragments produced throughout whole lifecycle. Traceability facilitates product and process management and control. Traceability is [9] a property of a system description technique that allows changes in one of the system descriptions – requirements specification, design, code, documentation, or test scenarios – to be traced to the corresponding fragments of the other descriptions. Further, such correspondence relationships should be maintained throughout the life time of a system in order to manage the artifact.

Traceability has received attention in the requirements engineering literature [8] [21], where change management requires special efforts because of the highly iterative process and frequent re-conceptualizations. Especially, the pre-requirements traceability has been studied thoroughly [13] [14]. However, there is a lack of traceability tools to support the full life-cycle, starting from artifact inception through for-
malization process to its use. Different representation formats that are used throughout the development process make it complicated to cover the whole life-cycle of an artifact. Given, a single requirement maps to multiple architectural and design concerns which is used to derive, it is difficult to maintain the consistency and traceability. Moreover, an architectural or design component has a number of other relations to various requirements. The task becomes even more difficult in the face of a large system that is built to satisfy thousands of requirements.

System specifications consist of a wide variety of fragments (artifacts), i.e. different kinds of information about system that together comprise a full (or partial) system specification at various levels of abstraction. Some of these artifacts are well structured, textual or graphical documents, while others are more loosely structured. In a geographically distributed project developers may use different tools to create and modify product fragments. The fragments can be refined iteratively and further processed by colleagues. Afterwards produced fragments are interchanged among members of a project, so that is important for colleagues to interpret an artifact correctly. The main challenges are to interrelate and manage all artifacts in different representation formats that are produced in a distributed manner using different tools, and to cover the whole product lifecycle.

The objective of this work is to present an approach to product fragments management during the distributed collaborative development process. The assumptions are that there are intra-related concepts in the problem domain and fragments are mapped to them. Given those conditions, the semantics of an artifact are increased by the artifact mapping to the corresponding concepts, and that enables predicting and assessing fragment change impact on other fragments.

The paper is structured as follows. In section two, related work is analyzed. In section three, the domain model based approach for product fragments management is presented. In section four, a case study is applied and illustrated by using weighted graphs. Finally, in section five, the work is concluded and its possible shortcomings with some insight to how to solve them are discussed.

2 Related Works

Over the recent years, a number of techniques have been proposed to facilitate management of product development through traceability enabling techniques. Some examples are [8] cross referencing schemes, based on some form of tagging, numbering, or indexing; and requirements traceability matrices. Studies in the field of traceability have mainly focused on specific parts of the development process [18] – mostly in the areas of pre-requirements traceability (e.g. [13] [14]) and linking requirements to architectural components (e.g. [10], [15]).

Some of the approaches are based on a specific modeling language and/or a tool. A much cited tool is TOOR (Traceability of Object-Oriented Requirements) [13], which is based on FOOPS, a formal object-oriented language. Integrating textual specifications and UML (Unified Modeling Language) model elements is used by Letelier [11], as a framework for configuring requirements traceability. Both approaches are restricted to FOOPS and UML respectively and can sequentially only be applied to software process based on the same language.