The Journal of Systems and Software 000 (2017) 1–23

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

CollabRDL: A language to coordinate collaborative reuse

Edson M. Lucas[a,b,*], Toacy C. Oliveira[a,d], Kleinner Farias[c], Paulo S.C. Alencar[d]

[a] PESC/COPPE, Federal University of Rio de Janeiro, Brazil
[b] IPRJ/UERJ, Polytechnic Institute, State University of Rio de Janeiro, Brazil
[c] PIPCA, University of Vale do Rio dos Sinos (Unisinos), Brazil
[d] David Cheriton School of Computer Science, University of Waterloo, Canada

ARTICLE INFO

Article history:
Received 19 June 2015
Revised 17 January 2017
Accepted 31 January 2017
Available online xxx

Keywords:
Software reuse
Collaboration
Framework
Language
Reuse process

ABSTRACT

Coordinating software reuse activities is a complex problem when considering collaborative software development. This is mainly motivated due to the difficulty in specifying how the artifacts and the knowledge produced in previous projects can be applied in future ones. In addition, modern software systems are developed in group working in separate geographical locations. Therefore, techniques to enrich collaboration on software development are important to improve quality and reduce costs. Unfortunately, the current literature fails to address this problem by overlooking existing reuse techniques. There are many reuse approaches proposed in academia and industry, including Framework Instantiation, Software Product Line, Transformation Chains, and Staged Configuration. But, the current approaches do not support the representation and implementation of collaborative instantiations that involve individual and group roles, the simultaneous performance of multiple activities, restrictions related to concurrency and synchronization of activities, and allocation of activities to reuse actors as a coordination mechanism. These limitations are the main reasons why the Reuse Description Language (RDL) is unable to promote collaborative reuse, i.e., those related to reuse activities in collaborative software development. To overcome these shortcomings, this work, therefore, proposes CollabRDL, a language to coordinate collaborative reuse by providing essential concepts and constructs for allowing group-based reuse activities. For this purpose, we extend RDL by introducing three new commands, including *role*, *parallel*, and *doparallel*. To evaluate CollabRDL we have conducted a case study in which developer groups performed reuse activities collaboratively to instantiate a mainstream Java framework. The results indicated that CollabRDL was able to represent critical workflow patterns, including parallel split pattern, synchronization pattern, multiple-choice pattern, role-based distribution pattern, and multiple instances with decision at runtime. Overall, we believe that the provision of a new language that supports group-based activities in framework instantiation can help enable software organizations to document their coordinated efforts and achieve the benefits of software mass customization with significantly less development time and effort.

1. Introduction

Modern software systems are developed by people working together, since the complexity of these systems requires knowledge related to numerous fields, including programming languages, human-machine interfaces and databases, which goes beyond the knowledge associated with a single system application domain. In this context, collaboration among people emerges as an important factor for the success of a software project development, and, therefore, tools to support collaborative work are crucially needed (Barthelmess and Anderson, 2002).

Another important aspect pertaining the construction of software systems is Software Reuse (Frakes and Kang, 2005). This concept involves, in part, reusing the knowledge acquired in previous projects during the development of a current project, and can result in higher-quality outcomes and resource savings. For this, some reuse techniques have proposed in the last decades, such as the RDL, a Reuse Description Language (Oliveira et al., 2007). In this scenario, Collaborative Software Reuse (Mendonça et al., 2008) (Noor et al., 2007) combines concepts of collaborative work and software development with those related to reusable artifacts, so that the development process can progress harmoniously (Mohagheghi and Conradi, 2007).

* Corresponding author.
E-mail addresses: edmlucas@cos.ufrj.br, emlucas@iprj.uerj.br (E.M. Lucas), toacy@cos.ufrj.br (T.C. Oliveira), kleinnerfarias@unisinos.br (K. Farias), palencar@uwaterloo.ca (P.S.C. Alencar).

http://dx.doi.org/10.1016/j.jss.2017.01.031
0164-1212/

In order to achieve the full potential of reusing software artifacts, one must embrace Systematic Reuse, which advocates the need of repeatable and (semi-)formal reuse processes (Rothenberger et al., 2003), where reusable artifacts and their associated constraints are known upfront Furthermore, reuse should be planned and coordinated (Malone and Crowston, 1994), and developers must perform the reuse activities configuring reusable artifacts in a collaborative and coordinated way to avoid errors and rework. A key aspect when practicing Systematic Reuse is documentation, and the documentation of a typical collaborative software reuse process needs to describe activities that can be interactive. Thus, multiple executions of the same process can produce different software behavior, as they are consequences of choices and responses resulting from interactive activities. Therefore, the documentation can support building software with different characteristics for the same domain, e.g., when a team decides to reuse the framework Portlet to build Web-based systems (Bellas, 2004) (Hepper, 2008).

In addition, Oliveira et al. propose a Reuse Description Language (RDL) for describing reuse processes and minimizing the problems associated with the instantiation of object-oriented frameworks (Oliveira et al., 2007). RDL is a textual and executable language, allowing the representation of reuse activities organized as a reuse process. RDL is also an interactive language. As a result, the RDL runtime environment prompts reusers during the framework instantiation process, to gather application-specific information (Oliveira et al., 2007, 2011).

Although RDL is effective for representing reuse activities, it falters when a collaborative reuse process is needed, a common scenario in software development projects. Today, a program in RDL expresses a sequential reuse process typically representing a single reuser, and is therefore unsuitable for complex reuse situations (Oliveira et al., 2007, 2011). The key problems are that RDL is (1) imprecise for specifying the interplay between the reuse activities, (2) inefficient for allowing developers to create working groups based on the available critical skills and responsibilities, and (3) ineffective for specifying how different working groups should perform distinct reuse activities collaboratively and in parallel. Hence, developers end up being unable to use the RDL constructs to support a systematic collaborative reuse process, especially when parallel activities performed by specific working groups need to be represented.

This paper, therefore, extends RDL towards supporting collaborative reuse activities. The extension, which leads to a new language called CollabRDL, involves the definition of three commands: (1) *Role* allows assigning reuse activities to working groups; (2) *Parallel* allows modularizing a set of commands that can be simultaneously performed; and (3) *Doparallel* allows performing blocks of commands concurrently. These commands were selected for our collaborative reuse extension for three reasons.

First, to promote a systematic, collaborative reuse in a coordinated way in RDL, developers must be able to allocate reuse activities to development team members considering their skills and responsibilities. Moreover, the current literature (e.g., (De Paoli and Tisato, 1994) (OASIS, 2006) (BPMN, 2011) (Cortes and Mishra, 1996) (Li and Muntz, 2000) (Briggs et al., 2003) (Fuks et al., 2007)) highlights that collaborative languages must allow associating activities to working groups; otherwise, the collaboration in development teams can be compromised. In fact, the Communication, Coordination and Cooperation (3C) model, proposed in (Ellis et al., 1991), refers to the activity-aware software development as a way to generate context for the execution of activities based on the understanding of activities performed by other developers.

Second, an ever-present need in collaborative software development is the concurrent execution of activities. For this, developers need to carefully define upfront which activities may be performed in parallel. Unfortunately, these definitions are usually done based on several mentally held indicators (a.k.a. experience) of developers, or even by personal communication (i.e., informally). This may transform the modularization of commands that can be run together into an informal but error-prone task.

Third, the current version of the RDL cannot determine how blocks of activities must be performed in parallel, despite its capacity to represent the sequential behavior, for example, using a traditional Loop command. Today, pivotal concepts (e.g., coordination) for supporting the simultaneity and synchronization of activities are still lacking. Hence, the RDL fails to reach the required coordination of modularized blocks of activities so as to enable them to work together effectively. Therefore, we argue that these commands are necessary and sufficient to support the broader collaboration issues found in RDL. We make also no claims about the generality of the proposed commands beyond collaboration in the context of RDL programs.

In addition, these new commands overcome critical problems, which include the inability of specifying how the produced artifacts and the acquired knowledge should be reused and the lack of constructs for describing how the reuse activities should be performed by groups of developers asynchronously or in parallel. These commands are fully supported by a runtime environment based on the workflow and Business Processes Manager (BPM), which provides facilities to load, start and run processes, besides allowing workflow functionalities. We chose Business Process Model and Notation (BPMN) to CollabRDL environment because BPMN is a pattern in the context of BPM that has been maintained by the Object Management Group (OMG). Furthermore, there are many environments offering support to BPMN (BPMN, 2016). Our initial evaluation has shown that the proposed commands are effective by using them to represent a set of well-established workflow patterns and performing a realistic case study in which two working groups collaboratively performed reuse activities for instantiating a mainstream Java framework, OpenSwing (OpenSwing, 2015). In total, the reuse process led to the creation of 49 attributes and the redefinition of 90 methods.

The remainder of this paper is organized as follows. Section 2 briefly introduces the concepts of collaboration, reuse process and describes RDL. Section 3 presents CollabRDL as an extension of RDL and describes its commands. Section 4 presents an environment for running reuse processes expressed in CollabRDL. Section 5 presents our evaluation of CollabRDL. Section 6 reviews related work and, lastly, Section 7 concludes our paper with a final discussion and a brief description of future work.

2. Background

CollabRDL aims at defining a collaborative reuse process. As a result, the following subsections briefly introduce the main concepts used in this work, such as Collaboration, Reuse Processes and RDL.

2.1. Collaboration

Collaboration is a kind of cooperation where interactions between people must take place in an organized and planned manner to achieve a common goal. Fuks et al. (2007) explore the 3C model that was originally presented by Ellis et al. (1991). The 3C model is composed of Communication, Coordination and Cooperation. Coordination makes the link between communication and cooperation in order to promote collaboration. Cooperation is a set of operations during a session in a shared working environment in the context of groupware (Ellis et al., 1991). In this sense, the 3C model emphasizes the importance of awareness, defined by

Please cite this article as: E.M. Lucas et al., CollabRDL: A language to coordinate collaborative reuse, The Journal of Systems and Software (2017), http://dx.doi.org/10.1016/j.jss.2017.01.031