Information and Software Technology 55 (2013) 126–152

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

Empirical evaluation of search based requirements interaction management

Yuanyuan Zhang *, Mark Harman, Soo Ling Lim

University College London, Malet Place, London WC1E 6BT, UK

ARTICLE INFO

Article history:
Available online 21 April 2012

Keywords:
Requirements
RIM
NSGA-II
Repair method
Dependency
Search-Based Software Engineering

ABSTRACT

Context: Requirements optimization has been widely studied in the Search Based Software Engineering (SBSE) literature. However, previous approaches have not handled requirement interactions, such as the dependencies that may exist between requirements, *and*, *or*, *precedence*, *cost*- and *value*-based constraints.
Objective: To introduce and evaluate a Multi-Objective Search Based Requirements Selection technique, using chromosome repair and to evaluate it on both synthetic and real world data sets, in order to assess its effectiveness and scalability. The paper extends and improves upon our previous conference paper on requirements interaction management.[1]
Method: The popular multi-objective evolutionary algorithm NSGA-II was used to produce baseline data for each data set in order to determine how many solutions on the Pareto front fail to meet five different requirement interaction constraints. The results for this baseline data are compared to those obtained using the archive based approach previously studied and the repair based approach introduced in this paper.
Results: The repair based approach was found to produce more solutions on the Pareto front and better convergence and diversity of results than the previously studied NSGA-II and archive-based NSGA-II approaches based on Kruskal–Wallis test in most cases. The repair based approach was also found to scale almost as well as the previous approach.
Conclusion: There is evidence to indicate that the repair based algorithm introduced in this paper is a suitable technique for extending previous work on requirements optimization to handle the requirement interaction constraints inherent in requirement interactions arising from dependencies, *and*, *or*, *precedence*, *cost*- and *value*-based constraints.

1. Introduction

In the release planning for software development, the requirements interdependency relationship is an important element which reflects how requirements interact with each other in a software system. This relationship can directly affect requirements selection activity as well as requirements traceability management, reuse and the evolution process.

According to Carlshamre et al.,

> "The task of finding an optimal selection of requirements for the next release of a software system is difficult as requirements may depend on each other in complex ways" [2].

Some requirements might have technical, structural or functional correlations that need to be fulfilled together or separately, or one requirement might be the prerequisite of another. The analysis and management of dependencies among requirements is called Requirements Interaction Management (RIM) which is defined as

> "the set of activities directed towards the discovery, management, and disposition of critical relationships among sets of requirements" [3].

Robinson et al. [3] defined requirements interaction as:

> "Two requirements R_1 and R_2 is said to interact if (and only if) the satisfaction of one requirement affects the satisfaction of the other."

RIM consists of a series of activities related to requirement dependencies which are complex and challenging tasks. The study is based on the assumption that the dependence identification activity has been completed. Here we present the most common interaction types found in the requirements literature.

* Corresponding author. Tel.: +44 (0) 20 7679 1056; fax: +44 (0) 20 3108 5040.
E-mail addresses: yuanyuan.zhang@cs.ucl.ac.uk (Y. Zhang), m.harman@cs.ucl.ac.uk (M. Harman), sooling.lim@cs.ucl.ac.uk (S.L. Lim).

[1] This paper is an invited extension of the previous conference paper that appeared at SSBSE 2010 [1]. The primary novelty of this paper over its predecessor is the introduction of the repair based approach and the extension of the evaluation to include the real world data set (RALIC) in addition to the synthetic data used in the conference version.

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.03.007

And	Given requirement R_1 is selected, then requirement R_2 has to be chosen
Or	Requirements R_1 and R_2 are conflicting to each other, only one of R_1, R_2 can be selected (exclusive OR)
Precedence	Requirement R_1 is selected before selecting requirement R_2
Value-related	Given requirement R_1 is selected, then this selection affects the value of requirement R_2 to the stakeholder
Cost-related	Given requirement R_1 is selected, then this selection affects the cost of implementing requirement R_2

Requirements dependency can have a very strong impact on the development process. For example, Bagnall et al. [4] considered the *Precedence* dependency, representing the relationship as a directed acyclic graph. Its vertices are denoted as individual requirements and its edges, directed from one vertex to another, are denoted as the *Precedence* dependency between the requirements. The authors showed that stakeholder subset selection process could be affected by *Precedence* dependency. Greer and Ruhe [5] considered *And* and *Precedence* dependencies. The proposed system provided candidate subsets of requirements for the next release problem. As with previous work, the authors use a single objective formulation, taking two type dependencies and the resource budget as the constraints. More recently, Tonella et al. [6] considered *Precedence* dependency in the requirements prioritization process. Similar to Bagnall et al. [4], *Precedence* relations were represented in a directly acyclic graph and treated as constraints for the ordering of the requirements.

And and *Precedence* dependencies were considered in the literature. However, little work has been studied on all five dependencies together. Proper treatment of RIM should take account of all the different types of requirement interactions. In this paper, we treat RIM problem as a constraint satisfaction problem and propose three search-based algorithms to handle all of the five common types of requirement dependencies for the first time. The objective is to investigate the influences of requirement dependencies on the automated requirements selection process for release planning.

Although search based techniques can find good solutions for unconstrained or simple constrained optimization problems, they might encounter difficulties while solving highly constrained problems. In terms of the RIM, the strength of constraints depends on the number and complexity of interactions between the requirements. The tighter the constraints are, the more difficult the problem is to solve.

The paper will show how multi-objective SBSE can be adapted to take account of RIM. In order to meet the challenge and to generate feasible optimal solutions, two improved techniques are used: one is an archive based version of NSGA-II; the other is a standard evolutionary algorithm with constraint handling technique – the 'repair' method. A real world large scale data set RALIC and the synthetic data sets previously studied [1] are adopted in this paper to evaluate the approach.

The rest of the paper is organized as follows. In Section 2 the problem is formalized as an SBSE problem. Section 3 describes the data sets, algorithms used and the performance metrics. Section 4 presents the results for dependence aware requirements optimization and discusses the findings. Section 5 describes the context of related work in which the current paper is located. Section 6 concludes the paper.

2. Problem formulation

In the context of Value/Cost-based requirements assignments analysis, the dependencies among requirements need to be accounted for within the fitness function. This section describes our fitness computation and how we incorporate RIM into this fitness.

Assume that the set of possible software requirements is denoted by:

$$\Re = \{r_1, \ldots, r_n\}$$

A set of stakeholders for a software system or service is denoted by $C = \{c_1, \ldots, c_m\}$. Each stakeholder may have a degree of importance for the company that can be reflected by a weight factor. The set of relative weights associated with each stakeholder c_j $(1 \leqslant j \leqslant m)$ is denoted by a weight set: $Weight = \{w_1, \ldots, w_m\}$ where $w_j \in [0, 1]$ and $\sum_{j=1}^{m} w_j = 1$.

The resources needed to implement a particular requirement can be transformed into cost terms. The resultant cost vector for the set of requirements $r_i (1 \leqslant i \leqslant n)$ is denoted by: $Cost = \{cost_1, \ldots, cost_n\}$.

In the real world, different stakeholders have different needs and perspectives. That is, not all requirements are equally important for a given stakeholder. Each stakeholder c_j $(1 \leqslant j \leqslant m)$ assigns a *value* to requirement r_i $(1 \leqslant i \leqslant n)$ denoted by: $v(r_i, c_j)$ where $v(r_i, c_j) > 0$ if stakeholder c_j desires implementation of the requirement r_i and 0 otherwise.

The overall *score* of a given requirement r_i $(1 \leqslant i \leqslant n)$ can be calculated by:

$$score_i = \sum_{j=1}^{m} w_j \cdot v(r_i, c_j) \tag{1}$$

The '*score*' of a given requirement is represented as its overall '*value*' for the company.

Next, we define the RIM constraints that were listed informally in the introduction to this paper.

And	Define a pair of requirements (i, j) and a set ξ such that $(i, j) \in \xi$ means that r_i is selected if and only if requirement r_j has to be chosen
Or	Define a pair of requirements (i, j) and a set φ such that $(i, j) \in \varphi$ (equivalently $(j, i) \in \varphi$) means that at most one of r_i, r_j can be selected
Precedence	Define a pair of requirements (i, j) and a set χ such that $(i, j) \in \chi$ means that requirement r_i has to be implemented before requirement r_j
Value-related	Define a pair of requirements (i, j) and a set ψ such that $(i, j) \in \psi$ means that if the requirement r_i is selected, then its inclusion affects the value of requirement r_j for the stakeholder
Cost-related	Define a pair of requirements (i, j) and a set ω on the requirements array R such that $(i, j) \in \omega$ means that if the requirement r_i is selected, then its inclusion affects the cost of implementing requirement r_j

In addition, the sets ξ, φ and χ should satisfy

$$\xi \bigcap \varphi = \emptyset \wedge \xi \bigcap \chi = \emptyset$$

in order to guarantee consistency in the requirements dependency relationship.

The fitness function with dependency constraints is defined as follows: