LAB 2:
Requirements Prioritization & Release Planning
Preparations and instructions

Björn Regnell and Oskar Prântare

October 19, 2014

1 Introduction

1.1 Purpose

This document provides instructions on how to prepare for and run a computer lab session on requirements selection. The lab session illustrates how requirements prioritization and release planning can be supported by computer tools, and demonstrates the complexity in finding solutions to these problems. *The preparations in Section 2 should be completed before the actual lab is run.*

1.2 Prerequisites

This lab assumes that you have installed the open source tool reqT.org and that you are familiar with basic requirements modeling using reqT. It is also assumed that you have completed Lab 1 Requirements Modeling.

1.3 Background

In this lab you will learn how to get started with requirements prioritization and release planning through the open source tool reqT, and reflect on how you could select requirements in your own project.

In real-world requirements engineering you are continuously faced with different types of trade-off problems. As we have limited development resources and normally would like our most important features to be ready as soon as possible, we need to make hard decisions on what to develop next and what to postpone. If we spend some time on assessing the cost and benefit of the tings we have at hand, we can hopefully find a good balance in how we spend our effort wisely in

relation to the available lead time. Two main trade-off problems in requirements engineering are

- **requirements prioritization**, where (a subset of) requirements are traded off against each other according to the opinions of the stakeholders based on some criteria such as the *benefit*, e.g. with respect to the strengthening of our product's brand, or the *cost*, e.g. of lost sales in case a requirement is not implemented. There are many prioritisation methods that can be used to elicit the stakeholders' opinions. In this lab we use the $100 method and an ordinal scale ranking method, and
- **release planning**, where requirements are scheduled in time over several releases under trade-offs with respect to constraints of the available capacities of different resources, requirements priorities and requirements interdependencies.

It is also likely that you will need to do hard choices of how you spend your efforts in the requirements engineering process. Probably you will have to make trade-offs such as: Is it more important to do more stakeholder analysis at this point, or should we instead focus on validation of the quality requirements that we have elicited so far? The prioritization planning methods in this lab may be used to prioritize and plan the tasks of the requirements process itself, although we will exemplify the methods by using features under consideration for development.

2 Preparations

Before doing the lab session in Section 3, complete all preparations in this section and bring requested items to the lab. In particular you need to make sure that you can access the text files you prepare below at your lab session computer.

2.1 Prioritization Preparations

2.1.1 Definitions

Here is one way to formalize the requirements prioritization problem:

S is a set of m stakeholders, $S = \{s_1, s_2..., s_m\}$

Q is set of n requirements $Q = \{q_1, q_2, ..., q_n\}$

$p(s_i, q_j)$ is a number representing the importance of requirement q_j assigned by stakeholder s_i

$w(s_i)$ is a number representing the importance of stakeholder s_i

$P(q_j)$ is the total priority of requirement q_j calculated by some function that maps all $p(s_i, q_j)$ and $w(s_i)$ to a single, numeric value.

A *prioritization method* defines a procedure that assigns numeric values to $p(s_i, q_j)$ for all stakeholders s_i and all requirements q_j, and to $w(s_i)$ for all stakeholders, according to some predefined priority criteria.

Before carrying out a prioritization method, a prioritization criteria needs to be defined. Examples of criteria are: market value, stakeholder benefit, risk of loss and urgency of delivery.

Define a prioritization criteria. Choose a prioritization criteria relevant to your project. Define your criteria so that it is desirable to maximize the priority value.

Criteria def.: _____

Define requirements and stakeholders. Make a reqT model with 2 stakeholders and 15 requirements from your project, analogous to this template:

```
Model(
  Req("autoSave"),
  Req("exportGraph"),
  Req("exportTable"),
  Req("modelTemplates"),
  Req("releasePlanning"),
  Req("syntaxColoring"),
  Req("autoCompletion"),
  Stakeholder("modeler"),
  Stakeholder("tester"))
```

Save the model in a text file called `req.scala`

2.1.2 Methods

The **$100 method** gives each stakeholder a fictitious sum of money to "spend" on the requirements, where $p(s_i, q_j)$ is assigned to the amount of money "spent" for each requirement representing its importance according to some criteria. The combined priorities $P(q_j)$ are calculated as

$$P(q_j) = \sum_{s_i \in S} p(s_i, q_j) w(s_i) k_i$$

where the normalization constants k_i are selected so that the sum of all $P(q_i)$ is normalized to 100 units, thus

$$k_i = \frac{100}{w \sum_{q_j \in Q} p(s_i, q_j)} \quad \text{where} \quad w = \sum_{s_i \in S} w(s_i)$$

Simplified $100 method. If all stakeholders are equally important, the formulas above can be simplified. Simplify $P(q_i)$ when $w(s_i) = a$ for all $s_i \in S$:

Use the $100 method. Put yourself in the shoes of each of your 2 stakeholders and use the $100 method to prioritize each of your 15 requirements according to your selected criteria.

Req	Id	Amount of dollars Stakeholder 1	Amount of dollars Stakeholder 2
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

Transfer your priority data above into a model file of this form:

```
Model(
  Stakeholder("modeler") has (
    Prio(1),
    Req("autoSave") has Benefit(25),
    Req("exportGraph") has Benefit(10),
    Req("exportTable") has Benefit(8),
    //...
    Req("autoCompletion") has Benefit(28)),
  Stakeholder("tester") has (
    Prio(2),
    Req("autoSave") has Benefit(3),
    Req("exportGraph") has Benefit(25),
    Req("exportTable") has Benefit(14),
    //...
    Req("autoCompletion") has Benefit(2)))
```

Save the model code in a text file called `prio100.scala`

The **ordinal priority ranking method** assigns a positive integer number to each requirement q_j for each stakeholder s_i denoted $p(s_i, q_j) \in [1..n]$, where n is the total number of requirements and all $p(s_i, q_j)$ are different for each stakeholder s_i. The priority $p(s_i, q_j)$ represents an ordinal scale estimation of the preference order of the requirement q_j according to the views of stakeholder s_i. Estimations on an ordinal scale imply that the estimates only provide ordinal information and not ratio information, which means that it is not possible to tell if one priority is, say 33% or 50% of another priority, just because it has a lower ordinal value.

One way to assign ordinal priority values is to use **pairwise comparison** of the requirements and then by some algorithm (e.g. insertion sort[1]) sort the requirements in priority order, and when the sorting is ready, let $p(s_i, q_j) = n$ for the first requirement, $p(s_i, q_j) = n - 1$ for the second, etc. down to $p(s_i, q_j) = 1$ for the last requirement.

If there are n requirements, what is the total number of possible pairwise combinations, without considering order?

Try these lines in the reqT console to check your answer above:

```
reqT> def allComb(n: Int) = (1 to n).combinations(2).toVector

reqT> allComb(100).foreach(println)

reqT> allComb(100).size    //size: _____
```

Consider a directed graph of comparisons, where a directed edge (a, b) represents a pair-wise comparison $a < b$. If there are n requirements nodes, what is the lowest number of comparison edges needed to connect all requirement with each other?

Try these lines in the reqT console to check your answer above:

```
reqT> def comb(n:Int) = (1 to n-1).map(i => (i,i+1)).toVector

reqT> comb(100).foreach(println)

reqT> comb(100).size //size: _____
```

[1] http://en.wikipedia.org/wiki/Insertion_sort

2.2 Release Planning Preparations

The reqT tool includes a constraint solver[2], enabling the formulation of prioritization and release planning problems using constraints over integer values in requirements models. After the problem has been formulated, then the constraint solver in reqT may automatically find a solution (if it exists), without the need for any further algorithm implementation.

[2] http://jacop.eu/

3 Lab Instructions

3.1 Prioritization Lab Instructions

3.2 Release Planning Lab Instructions

4 Conclusion and reflection

Reflect upon these discussion questions and write down some reflections:

1. How ... ?