# Objective

The risk-based test strategy suggests splitting overall new feature testing work into items, measure risk and test execution time associated with each item, and select resulting test coverage in such a way that the highest possible volume of risk is covered by test in budget of time available to test org for testing of that new feature. Depending on implementation of the risk-based testing approach and depending on a project stage on which it is involved, new features, feature functional requirements, or test cases can be used as items. Risk estimates associated with each item, and estimation of time required to cover that item, are out of scope of this work.

Our objective is to build a software tool that automates process of selection of items for execution based on existing risk factor and time budget associated with each item.

*Overall objective of a tool at a high level can be described by the knapsack problem.*

*The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. (see Wikipedia).*

*In our case, “weight” is an estimate of execution time, “value” is a risk level associated with each item, and “given limit” is an overall time budget available for handling set of items.*

RBTCS tool provides the ability to input data as an excel file, and generates output as an excel file containing list of items to be handled by test.

# High-level Functional Requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Status** |
| 1 | Input data as an excel file containing list of items and associated risk factor and execution time | Approved |
| 2 | Output data as an excel file containing list of selected items | Approved |
| 3 | Support command line arguments (filename, names of columns, available time budget) | Approved |
| 4 | Implement optimal algorithm for item selection: use dynamic programing approach for 0/1 knapsack problem:   * O(NW) for time complexity; * O(NW) for space complexity * N – is the number of items; * W – is the available time budget; | Approved |
| 5 | Implement sub-optimal algorithm if complexity for optimal is too high: use greedy algorithm:   * O(NlogN) time complexity; * O(N) space complexity; | Approved |
| 6 | Implement logging | Approved |
|  |  |  |

# Use Cases

The main use case for the tool is as follows:

1. User prepares input file in excel. Input file contains columns with estimated risk-factor and execution time for each item.
2. User run the tool, and in command line arguments specifies input file name, column names containing risk factor, execution time and column name where to put selection marks. User also provides overall time budget.
3. Tool produces output excel file which contains same data but with selection column filled (i.e. with data that shows which items have been selected for coverage, and which items haven’t been selected. Tool also shows risk coverage.

# Functional Specification

## Prerequisite requirements

* Python 2.7
* Python libraries (use “pip install <libname>” to resolve)
  + xlrd
  + xlwt
  + argparse
  + logging
  + enum

## Input data file specification

Input data file can either be .xlsx or .xls format.

Revised requirements:

Input data file must contain a table with following requirements:

* Data should be on the first sheet in the input file (other sheets are not processed);
* There should be a header row with column names. This can’t be the last row on a sheet. (This row will be detected automatically: tool will search for “--risk-factor”, “--execution-time” and “--selection” cells in a common row).
* The tool interprets remaining rows in the sheet as row containing items.
* Risk factor column content below header row should be float data type (or convertible to float).
* Execution time column content below header row should be integer data type (or convertible to integer).

## Command line arguments

Command line format:

python rbtcs.py [-h] [--risk-factor RISK\_FACTOR] [--execution-time EXECUTION\_TIME] [--selection SELECTION] [--time-budget TIME\_BUDGET] filename

Mandatory arguments:

* filename – specifies name of input file (either absolute path and name, or relative)

Optional arguments:

* -h – show help
* -r RISK\_FACTOR - specifies name of risk-factor column as RISK\_FACTOR value. Default value is “Risk Values”.
* -t EXECUTION\_TIME – specifies name of execution-time column as EXECUTION\_TIME value. Default value is “EXECost (MH)”.
* -s SELECTION – specifies name of selection column as SELECTION value. Default value is “Covered (n)?”
* -b TIME\_BUDGET – specifies overall time budget as TIME\_BUDGET value. Default value is 2500.
* -p PRECONDITION – specifies name of preconditions column. Default usage suggest no precondition relationship between items.

## Input data validations

The tool does following validations for input data:

* argument parsing of input arguments and generation of error messages in case if arguments don’t match specification. Error messages are defined in argparse library.
* Input filename validation (if the path is valid and file exists).
* Validation of a header row existence in an input file (i.e. a row which contains cells with risk factor, execution time and selection column names).
* Validation for time budget value to be positive integer number.
* Validation of values in risk factor column to be float type (or convertible to float).
* Validation of values in execution time column to be integer values (or convertible to integer).
* Comparison of time budget and number of items with threshold values to predict possible switchover to suboptimal algorithm.

## Selecting items for coverage

Tool first tries to use optimal algorithm based on 0/1 knapsack problem resolution. It guarantees optimal solution, i.e. a solution which produces such a set of items that has highest possible risk coverage under specified budget of time. The algorithm has O(NW) time complexity and space complexity, where N is number of items in seed file, and W is the specified time budget. In such conditions space complexity becomes the limiting factor.

In case if there are not enough resources to handle the task with optimal algorithm (it happens when tool catches memory exception while requesting memory for data structures), tool applies suboptimal algorithm based on greedy method. It sorts the items in decreasing order of risk per execution time, and selects as many items as possible with specified budget going from highest to lowest ratios.

## Output

Output file is always “rbtcs\_result.xls” file. It contains your input data with selection column filled with 0 and 1 values:

* “n” – this item is not selected for coverage;
* “y” – this item is selected for coverage.

In addition, the tool specifies the overall level of risk covered by selected items in output stream. This is floating value from 0 to 1.

## Logging messages

Current version has logging level set to INFO.

All log messages have the following format:

<date> <time> - rbtcs - <log level> - <log message>

For example:

2017-09-06 11:08:06,812 - rbtcs - CRITICAL - illegal input file name or file doesn't exist

There are 6 log levels:

* CRITICAL
* ERROR
* WARNING
* INFO
* DEBUG
* NOTSET

Logging messages:

|  |  |  |
| --- | --- | --- |
| **LEVEL** | **Message** | **Conditions** |
| CRITICAL | “%s is illegal input file name or file doesn't exist” | When %s argument specified in command line as a source file name doesn’t exist or represent illegal name. |
| CRITICAL | "Header row not found!" | Message generated when header row wasn’t found in an input file. I.e. there is no such row exists which contain cells with names specified in --risk-factor, --execution-time, --selection. |
| CRITICAL | "Header row can't be the last row in the file!" | Message generate when header row is the last row on an input sheet. |
| DEBUG | “Header row index: %d” | Message generated when header row detected. %d specifies the input file row number corresponding to header row. |
| DEBUG | “Risk Factor column index: %d” | Message generated when header row detected. %d specifies the input file column number corresponding to risk factor column. |
| DEBUG | “Execution Time column index: %d” | Message generated when header row detected. %d specifies the input file column number corresponding to execution time column. |
| DEBUG | "Selection column index: %d" | Message generated when header row detected. %d specifies the input file column number corresponding to selection column. |
| CRITICAL | “Time budget is not a positive number: %d” | Message is generated when a value of time budget specified in --time-budget TIME\_BUDGET argument is not a positive number. %d specifies budget provided. |
| CRITICAL | “Can't convert Risk Factor for item in row # %d to float” | Message appears when the tool is unable to convert value of risk factor for test cases with index %d into float. %d here specifies the item row number in input file. |
| CRITICAL | “Can't convert Execution Time for item # %d to integer” | Message appears when the tool is unable to convert value of execution time for test cases with index %d into integer. %d here specifies the item row number in input file. |
| WARNING | “Specified time budget is relatively big which may prevent from getting optimal solution” | Message is generated when specified time budget exceeds internal threshold MAX\_BUDGET (currently it is equal to 10000). Such a big budget may prevent from getting optimal solution due to potentially high space complexity. |
| WARNING | “Number of test cases in seed file is relatively big which may prevent from getting optimal solution” | Message is generated when number of items in the input file exceeds internal threshold MAX\_TC (currently it is equal to 300). Such a high number of items may prevent from getting optimal solution due to potentially high space complexity. |
| CRITICAL | “Error reading input file in XLRD” | Message is generated when there was an error in XLRD library while reading input file. (note: XLRD library is used to read input .xls and .xlsx files). |
| DEBUG | “XLRD Exception: <exception string>" | Message is generated when there was an error in XLRD library while reading input file. And it contains exception string returned by XLRD library for debug purposes. |
| INFO | “Building test coverage using optimal algorithm” | Message appears when the tool tries to calculate optimal solution for input data (which has been read and validated successfully). |
| INFO | “Covered risk with proposed test set using optimal algorithms is %f” | Message appears if optimal algorithm succeeded. %f value is a value of risk covered by selected items. |
| ERROR | “Caught MemoryError exception while building test set using dynamic programming algorithm for 01 knapsack problem” | Message appears when space complexity of optimal algorithm is it too big and led to memory exception. |
| INFO | “Building test coverage using greedy approximation algorithm” | Message specifies that greedy algorithm was triggered after optimal algorithm generated memory exception. |
| INFO | “Covered risk with proposed test set using greedy method is %f” | Message appears when greedy algorithm completed. %f value is a value of risk covered by selected items. |