Using the RelKit Application to Implement the LRM Process

Andrew Rowland
ReliaQual Associates, LLC

Contents

Chapter 1

Overview

The RelKit application is a software toolkit that can be used to help manage the Lifecycle Reliability Management (LRM) process. RelKit can be used to assist in the following LRM tasks:

- Warranty Analysis
- Reliability Risk Analysis
- Reliability Capability Analysis
- Reliability Risk Indexing
- Reliability Growth Planning
- Reliability Growth Tracking

1.1 RelKit Software

This section provides a brief description of the RelKit software.

1.2 Definitions

The following is a list of abbreviations and terms that are used throughout this manual.

1. **LRM** - Lifecycle Reliability Management
2. **RelKit Program** - this is a RelKit database.

Chapter 2

Creating a New RelKit Project

To use the RelKit software for managing LRM projects, you will need to create a RelKit Program. After creating a new RelKit Program, you will need to populate it with system hardware and warranty claim information. Creating a new RelKit Program for a LRM project requires the following steps, at minimum:

1. Select 'New Program' from either the toolbar or the *File* menu.
2. Either:
   (a) Manually create the system hardware structure.
   (b) Import the system hardware structure from an external file.
3. Either:
   (a) Manually enter all of the warranty claim information.
   (b) Import all of the warranty claim information from an external file.

2.1 Manually Building the Hardware Structure

To manually build the hardware structure, you will use the "Add Sibling Assembly" (Figure 2.1a) and "Add Child Assembly" (Figure 2.1b) tool buttons found on the RelKit Workbench tool bar. All new assemblies are added with reference to the currently selected hardware item. Thus, with a new RelKit Program, only the top-level assembly will be available.

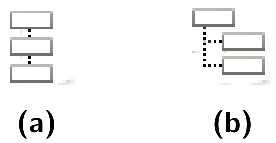

(a) (b)

Figure 2.1: Add Hardware Tool Buttons

Selecting to add a sibling assembly will result in the dialog in Figure **??** being displayed. In this dialog, enter the number of sibling assemblies you wish to add and press the 'OK' button. This will add the specified number of assemblies at the same indenture level as the selected hardware item. Note that RelKit will not allow you to add a sibling assembly to the top-level hardware item. Adding child assemblies occurs in a similar manner.

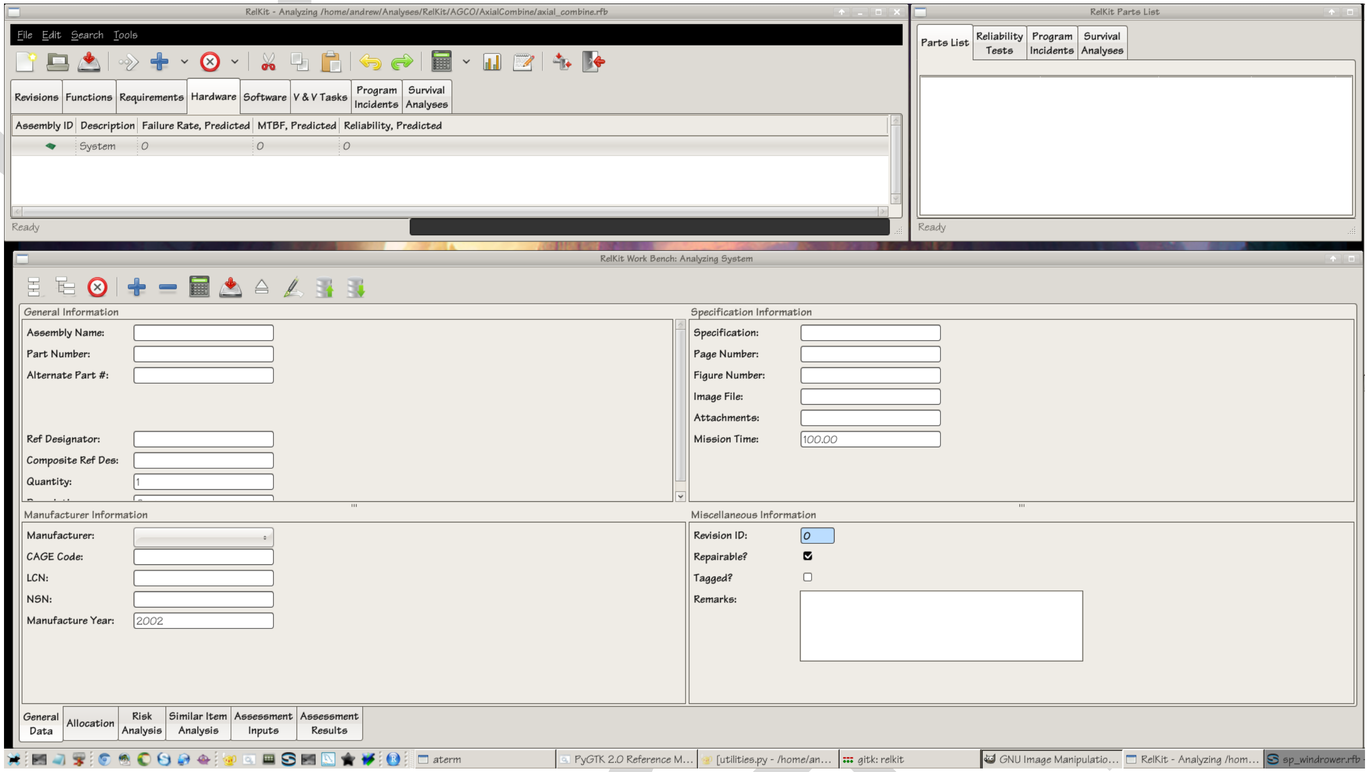
Once the hardware structure is constructed through the use of the "Add Sibling Assembly" and "Add Child Assembly" tool buttons, additional information such as the description or the name of each assembly must be manually entered. This is accomplished by first selecting the hardware item you wish to edit. With the hardware item selected, information related to the item can be added or edited in the RelKit Workbench window.

2.2 Importing the Hardware Structure

Figure 2.2 shows RelKit with the newly created RelKit Program open and selected to the Hardware tree tab. With a new project, there is only one line in the Hardware tree for the top-level assembly, or system. At this point it is necessary to create the remainder of the product structure. This can be done either by manually creating all of the remaining structure or by importing the information from an external file.

Figure 2.3 shows the assistant used to import the hardware structure. Figure 2.3a shows the dialog used to select the source file. Matching the source file columns to the appropriate database fields is done with the dialog shown in Figure 2.3b. A source column may be assigned to multiple database fields. It is recommended that the Description and Name fields in the database be mapped to the same source file column unless the source file has a unique column for each.

Finally, Figure 2.4 shows the Hardware Tree after the import. Note the default name of the top-level assembly has been updated. All fields for existing assemblies will be updated with new information if available. Also, we can two of the assemblies subordinate to the top-level assembly.



Assembly ID	Description	Failure Rate, Predicted	MTBF, Predicted	Reliability, Predicted
	System	0	0	0

General Information

Assembly Name:
Part Number:
Alternate Part #:
Ref Designator:
Composite Ref Des:
Quantity: 1

Specification Information

Specification:
Page Number:
Figure Number:
Image File:
Attachments:
Mission Time: 100.00

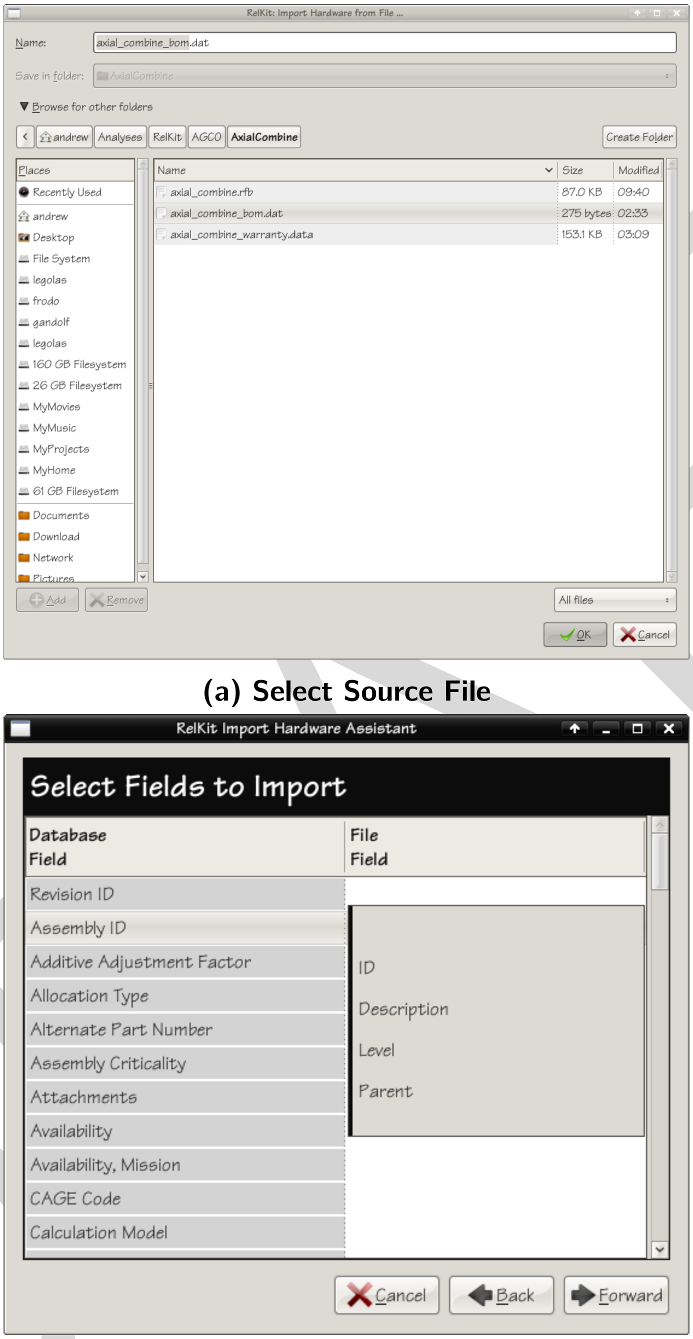
Manufacturer Information

Manufacturer:
CAGE Code:
LCN:
NSN:
Manufacture Year: 2002

Miscellaneous Information

Revision ID: 0
Repairable? ☑
Tagged? ☐
Remarks:

Figure 2.2: New RelKit Project



Name: axial_combine_bom.dat

Save in folder: AxialCombine

Name	Size	Modified
axial_combine.rfb	87.0 KB	09:40
axial_combine_bom.dat	275 bytes	02:33
axial_combine_warranty.data	153.1 KB	03:09

All files

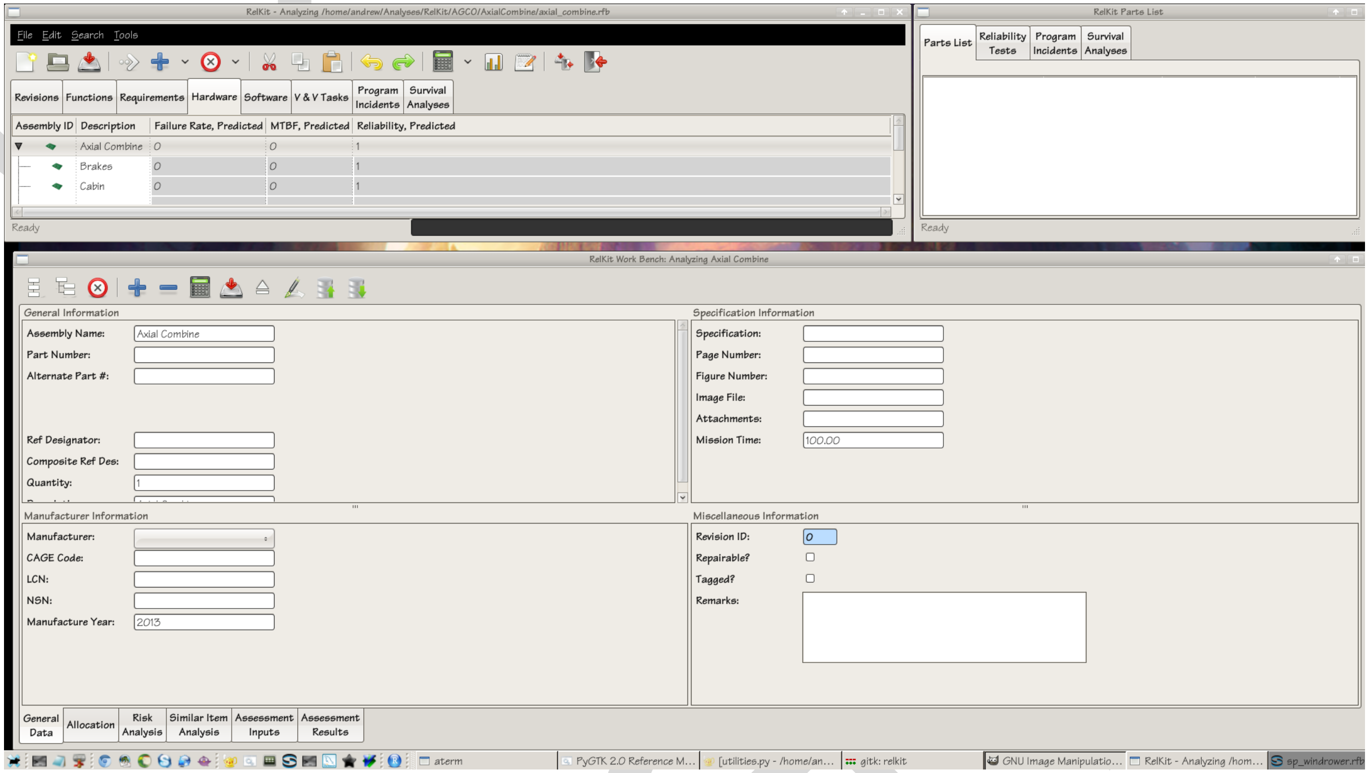
(a) Select Source File

Select Fields to Import

Database Field	File Field
Revision ID	
Assembly ID	
Additive Adjustment Factor	
Allocation Type	
Alternate Part Number	
Assembly Criticality	
Attachments	
Availability	
Availability, Mission	
CAGE Code	
Calculation Model	

(b) Assign Source Columns to Database Fields

Figure 2.3: Importing Hardware Structure to RelKit



RelKit - Analyzing /home/andrew/Analyses/RelKit/AGCO/AxialCombine/axial_combine.rfb

Assembly ID	Description	Failure Rate, Predicted	MTBF, Predicted	Reliability, Predicted
	Axial Combine	0	0	1
	Brakes	0	0	1
	Cabin	0	0	1

Ready

RelKit Parts List

Ready

RelKit Work Bench: Analyzing Axial Combine

General Information

Assembly Name:	Axial Combine
Part Number:	
Alternate Part #:	
Ref Designator:	
Composite Ref Des:	
Quantity:	1

Specification Information

Specification:	
Page Number:	
Figure Number:	
Image File:	
Attachments:	
Mission Time:	100.00

Manufacturer Information

Manufacturer:	
CAGE Code:	
LCN:	
NSN:	
Manufacture Year:	2013

Miscellaneous Information

Revision ID:	0
Repairable?	☐
Tagged?	☐
Remarks:	

Figure 2.4: Hardware Tree After Importing