

A Formal Requirements Engineering Toolkit

B. Eng. Project Example

Oliver Dixon

Department of Computer Science, University of York

Saturday, 20th June 2026

<https://github.com/oliverdixon/OptiFOL-Software>

Research Question

- Executing large engineering projects is difficult and error-prone, even for experienced professionals.
- Attaining a clear set of *project requirements* from the outset has been shown, over decades, to improve outcomes.
- But how do we know that our requirements are *useful*?
 - More precisely, how do we specify them?
 - And how do we know that other people know what we mean?
 - And if we have 100,000 requirements, how do we know that no two requirements are contradictory?
- These questions are hard to answer.
- So can we develop some software to make the process easier?

Research Answer

- We can use the language of *First-Order Logic* to formally represent requirements with no ambiguity.

“The submarines should not be detectable by any sensors”:

$$\underbrace{\forall x(\text{Submarine}(x))}_{\text{for all submarines...}} \implies \underbrace{\neg \exists y(\text{Sensor}(y) \wedge \text{CanDetect}(y, x))}_{\text{there is no sensor... that can detect our sub!}}$$

- Once we have our requirements in first-order logic, we can reason about them using techniques from *Automated Theorem-Proving*.
 - Can they contradict one another?
 - Are any of them redundant?
 - Can they be made any simpler?

Project Outcomes

- Wrote approx. 35,000 lines of C++ to *prove* that this can be done.
- Produced a 61-page report (frontmatter; 30 pages of body; 15 pages of appendices; bibliography) describing the problem specification, design, implementation, and evaluation.
- Project marked at 81% by a team of world-leading experts in mathematical logic.
- Software has attracted interest from industrial partners.

Optifol_



Inspiring
Confidence in
Engineering

Visualising the Proof Procedure

Optifol

File Edit Help

Requirements Index Analysis and Optimisation Testing and Compliance Releases and Reports

Search by Name

Requirement	Statement	Normalised Statement
▼ Unassigned Requirements		
AnimalLovers	$\text{ForAll } x (\text{ForAll } y (A(y) \Rightarrow L(x, y)) \Rightarrow \text{ThereExists } y L(y, x))$	$\{ \{ A(S0(x\#0)), L(S1(x\#0), x\#0) \}, \{ \neg L(x\#0, S0(x\#0)), L(S1(x\#0), x\#0) \} \}$
AnimalKillers	$\text{ForAll } x (\text{ThereExists } z (A(z) \ \& \ K(x, z)) \Rightarrow \text{ForAll } y \neg L(y, x))$	$\{ \{ \neg A(z\#1), \neg K(x\#1, z\#1), \neg L(y\#1, x\#1) \} \}$
JackLovesAnimals	$\text{ForAll } x (A(x) \Rightarrow L(\#J, x))$	$\{ \{ \neg A(x\#2), L(\#J, x\#2) \} \}$
CatsAreAnimals	$\text{ForAll } x (C(x) \Rightarrow A(x))$	$\{ \{ A(x\#3), \neg C(x\#3) \} \}$
TunaCat	$C(\#T)$	$\{ \{ C(\#T) \} \}$
TunaKilled	$(K(\#J, \#T) \mid K(\#C, \#T))$	$\{ \{ K(\#C, \#T), K(\#J, \#T) \} \}$

new query

Ready.

Specifying a Requirement in Logic

The screenshot shows the Optifol application interface. The main window displays a table of requirements with columns for Name, Description, FOL Sentence, and Software Tests. A pop-up window provides details for the requirement 'AnimalLovers'.

Requirement Name: AnimalLovers

Description: Everybody who loves all animals is themselves loved by somebody.

FOL Sentence: $\forall x (\forall y (A(y) \Rightarrow L(x, y)) \Rightarrow \exists s)$

Software Tests: 1 test defined

Target Executable	Test Fixture	Test Name
/home/owd/CLionProjec	CKCPropertiesTests	AreAnimalLoversLoved

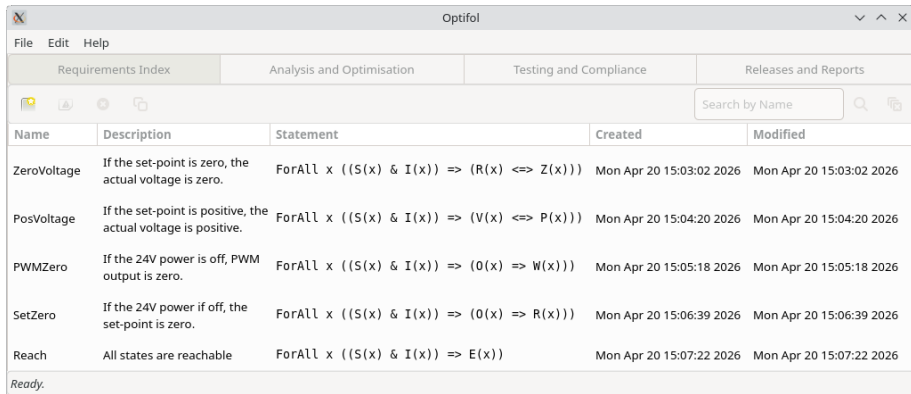
- AreAnimalLoversLoved ✓
- AreAnimalKillersHated
- AreCatsAnimals

Organising Requirements by Subsystem

The screenshot shows the Optifol software interface. The window title is "Optifol". The menu bar includes "File", "Edit", and "Help". The main area is divided into several panes. On the left is the "Project Explorer" showing a tree view of requirements. The main pane displays a "Requirements Index" table with columns for Name, Description, Statement, and Created. The table contains two entries: "AllGreeksAreMen" and "SocratesGreek".

Name	Description	Statement	Created
AllGreeksAreMen	Affirming that all Greek individuals are also humans.	$\text{ForAll } x (P(x) \Rightarrow Q(x))$	Wed Apr 1 14:38:15 2026
SocratesGreek	Socrates was a Greek	$P(\#S)$	Wed Apr 1 14:40:41 2026

Reviewing System Behaviour



The screenshot shows the Optifol software window with a menu bar (File, Edit, Help) and a toolbar. Below the toolbar is a navigation pane with tabs for 'Requirements Index', 'Analysis and Optimisation', 'Testing and Compliance', and 'Releases and Reports'. The 'Requirements Index' tab is active, displaying a table of requirements. The table has columns for Name, Description, Statement, Created, and Modified. The requirements listed are ZeroVoltage, PosVoltage, PWMZero, SetZero, and Reach. At the bottom of the window, the status bar shows 'Ready.'

Name	Description	Statement	Created	Modified
ZeroVoltage	If the set-point is zero, the actual voltage is zero.	$\text{ForAll } x \ ((S(x) \ \& \ I(x)) \Rightarrow (R(x) \ \Leftrightarrow \ Z(x)))$	Mon Apr 20 15:03:02 2026	Mon Apr 20 15:03:02 2026
PosVoltage	If the set-point is positive, the actual voltage is positive.	$\text{ForAll } x \ ((S(x) \ \& \ I(x)) \Rightarrow (V(x) \ \Leftrightarrow \ P(x)))$	Mon Apr 20 15:04:20 2026	Mon Apr 20 15:04:20 2026
PWMZero	If the 24V power is off, PWM output is zero.	$\text{ForAll } x \ ((S(x) \ \& \ I(x)) \Rightarrow (0(x) \ \Rightarrow \ W(x)))$	Mon Apr 20 15:05:18 2026	Mon Apr 20 15:05:18 2026
SetZero	If the 24V power is off, the set-point is zero.	$\text{ForAll } x \ ((S(x) \ \& \ I(x)) \Rightarrow (0(x) \ \Rightarrow \ R(x)))$	Mon Apr 20 15:06:39 2026	Mon Apr 20 15:06:39 2026
Reach	All states are reachable	$\text{ForAll } x \ ((S(x) \ \& \ I(x)) \Rightarrow E(x))$	Mon Apr 20 15:07:22 2026	Mon Apr 20 15:07:22 2026

Ready.

Adding Tests & Generating Qualification Reports

The screenshot shows the 'Optifol' application window with the 'Requirements Index' tab active. A requirement named 'AnimalLovers' is selected, and its details are shown in a pop-up window. The details include:

- Requirement Name:** AnimalLovers
- Description:** Everybody who loves all animals is themselves loved by somebody.
- FOL Sentence:** $\forall x(\forall y(A(y) \Rightarrow L(x,y)) \Rightarrow \exists l)$
- Software Tests:** 1 test defined

Below the requirement details, a table lists the tests defined for this requirement:

Target Executable	Test Fixture	Test Name
/home/owd/CLionProjec	CKCPropertiesTests	AreAnimalLoversLoved
		AreAnimalLoversLoved ✓
		AreAnimalKillersHated
		AreCatsAnimals

The screenshot shows the 'Optifol' application window with the 'Requirements Index' tab active. The 'Output Directory' is set to '/home/owd/CLionProjec'. The output of a test run is displayed in a text area:

```
Rc files read:
/etc/LaTeXMk
Latexmk: This is LaTeXmk, John Collins, 15
Latexmk: applying rule 'pdflatex'...
Rule 'pdflatex': Reasons for rerun
Changed files or newly in use/created:
/home/owd/CLionProjects/0FTestDemo/report
/home/owd/CLionProjects/0FTestDemo/report
Category 'changed user':
/home/owd/CLionProjects/0FTestDemo/report
/home/owd/CLionProjects/0FTestDemo/report

-----
Run number 1 of rule 'pdflatex'
-----
Running 'pdflatex -interaction=nonstopmode
-----
This is pdfTeX, Version 3.141592653-2.6-1.4
restricted BETA enabled
```