

Integration and Testing of Safety Case Tool Support

Sarah Bass

San Jose State University
San Jose, CA 95112, USA

Ewen Denney and Josef Pohl (Mentors)
SGT / NASA Ames Research Center
Moffett Field, CA 94035, USA

Abstract

Safety cases are a structured form of an argument that illustrate the safety of a system through the connections between claims and their evidence or justification. Ideally, they should provide for every possible instance in regards to safety of a system, and give proof through justifications and evidence that a system is indeed safe. The alternative to a safety case involves following a set of objectives that have been previously founded on standards or guidelines. Thus by following these objectives, the system or artifact is implicitly deemed as safe. The point of a safety case is to clearly demonstrate the safety of a system by going through thorough argumentation. The AdvoCATE tool provides support for the modeling and transformation of safety cases. Since conventional safety processes are based on spreadsheets, we want to integrate our tool with spreadsheet based artifacts, and then test it. In order to test the tool, we will be developing a test suite to assess functionality of the interface and transformations, so that it can be used for safety-critical applications.

Traditional Safety Processes Vs. Safety Cases

- Originally, safety verification procedures followed implicit argumentation, where systems were deemed safe when the standards and guidelines were followed.
- Alternatively, safety cases demonstrate explicit argumentation in terms of safety, going through each claim and evidence/justification.
- Traditional approaches to safety assurance record artifacts in tabular form, whereas safety cases can be presented in a graphical form.
- We have been working to develop a tabular model for our graphical data. The resulting table (Figure 2) provides a sample formatted version of the diagram

Background and Notation of Safety Cases

•A safety case is an evidence-based structured form of an argument to illustrate the safety of a system through claims and justifications. For example, Figure 1 is a safety case which argues that the auto pilot module accurately calculates the correct angle of attack and is justified by claims and evidence.

Elements and Notation:

- The notation used is Goal Structure Notation (GSN)
- Each structure represents a part of an argument (i.e. a claim or a strategy)
- Goal structures are: goal (claim): □ strategy(method to decompose goal): ▽
- context: □ evidence: ○ justification: ○ J assumption: ○ A
- Link structures are: IsSolvedBy: → InContextOf: →

AdvoCATE

- AdvoCATE is the safety case transformation system.
- It allows for the application of transformations (i.e. creating CSV files, narratives, or computing metrics) to safety cases
- Edits to safety cases can also be made using this tool.
- Currently, we are working on implementing a Microsoft Excel Macro with the tool to format the outputted CSV file into an organized spreadsheet (like the one below in Fig. 2).

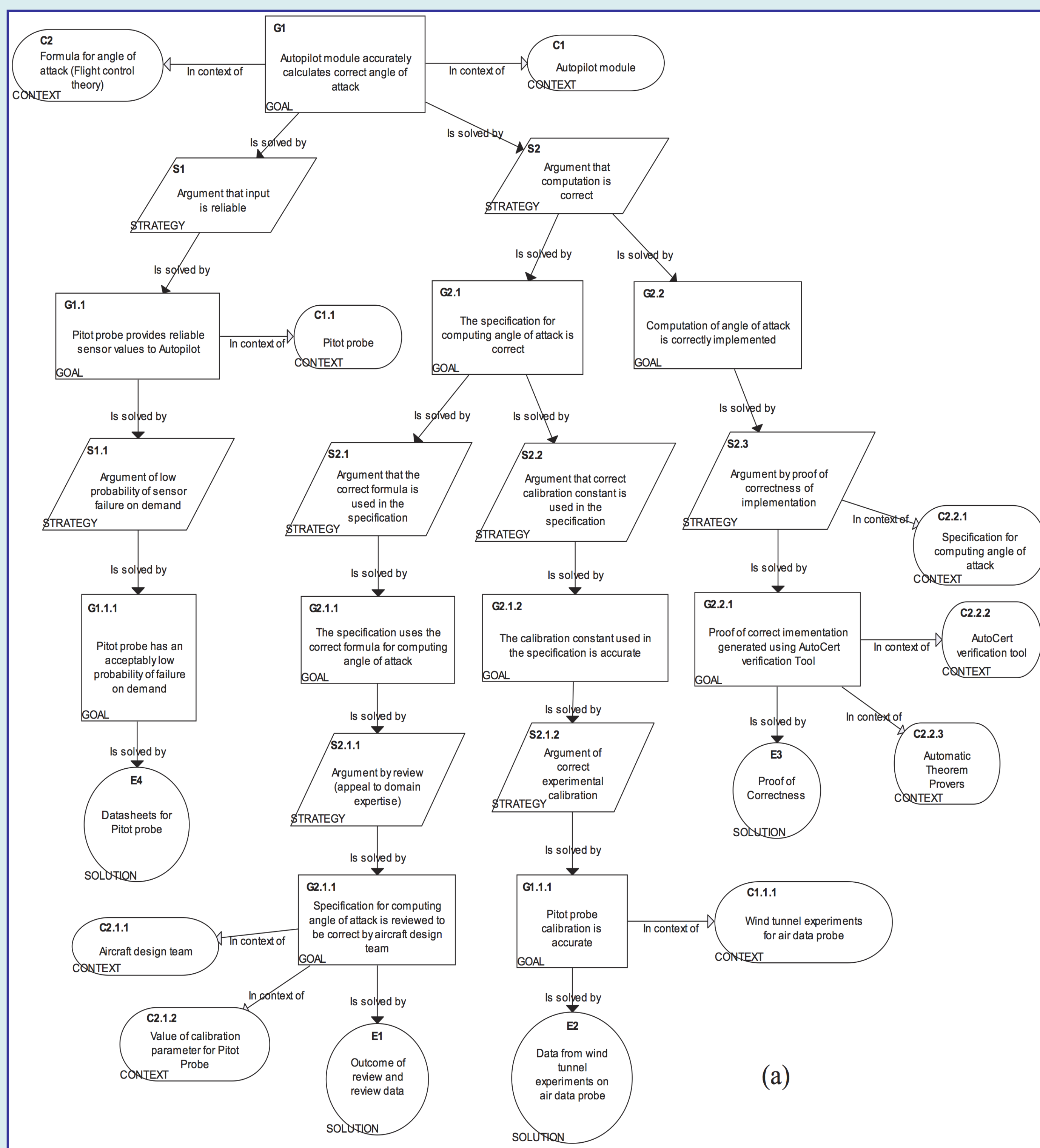


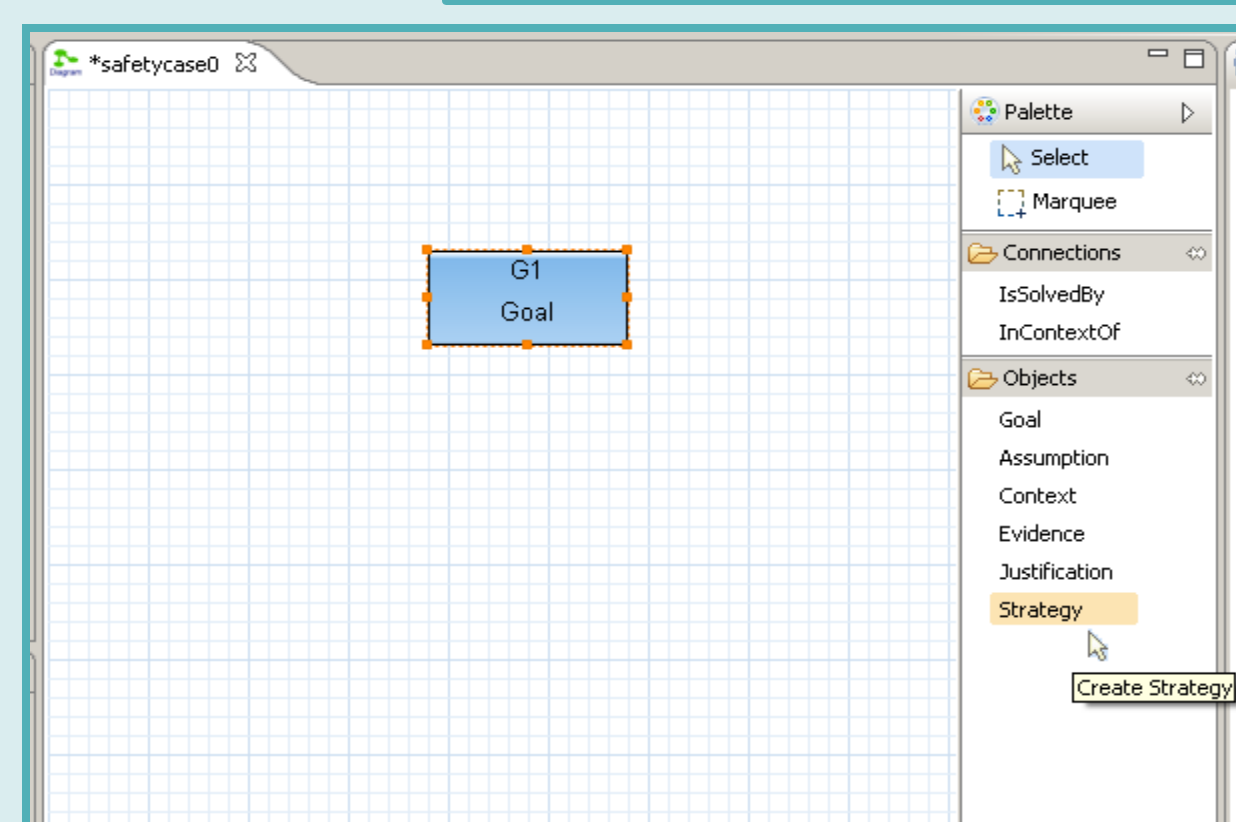
Figure 1. A Sample Safety Case Diagram using Goal Standard Notation (GSN)
Denney, E., Habli, I., Pai, G.: Perspectives on Software Safety Case Development for Unmanned Aircraft. In: Proc. 42nd Annual IEEE/IFIP Intl. Conf. on Dependable Sys. and Networks. (Jun 2012)

PARENT GOAL	CONTEXT	STRATEGY	SUBGOAL/SOLUTION
G1: Autopilot module accurately calculates correct angle of attack	C2: Formula for angle of attack (Flight control theory) C1: Autopilot module	Strategy Type S2: Argument that computation is correct	G2: Computation of angle of attack is correctly implemented G2.1: The specification for computing angle of attack is correct G1: Pitot probe provides reliable sensor values to Autopilot
G2.2: Computation of angle of attack is correctly implemented	C2.2.3: Automatic Theorem Provers C2.2.2: AutoCert verification tool	S2.3: Argument by proof of correctness of implementation	G2.2.1: Proof of correct implementation generated using AutoCert verification Tool E3: Proof of Correctness
G2.1: The specification for computing angle of attack is correct		S2.2: Argument that correct calibration constant is used in the specification S2.1: Argument that the correct formula is used in the specification	G2.1.2: The calibration constant used in the specification is accurate G2.1.1: The specification uses the correct formula for computing angle of attack G1.1: Pitot probe calibration is accurate
G2.1.1: The specification uses the correct formula for computing angle of attack	C1.1.1: Wind tunnel experiments for air data probe	S2.1.1: Argument by review (appeal to domain expertise)	E2: Data from wind tunnel experiments on air data probe
G8: Specification for computing angle of attack is reviewed to be correct by aircraft design team	C2.1.1: Aircraft design team C2.1.2: Value of calibration parameter for Pitot Probe	S2.1.1: Argument by review (appeal to domain expertise)	G8: Specification for computing angle of attack is reviewed to be correct by aircraft design team E1: Outcome of review and review data
G1.1: Pitot probe provides reliable sensor values to Autopilot	C1.1: Pitot probe	S1.1: Argument of low probability of sensor failure on demand	G1.1.1: Pitot probe has an acceptably low probability of failure on demand E4: Datasheets for Pitot probe

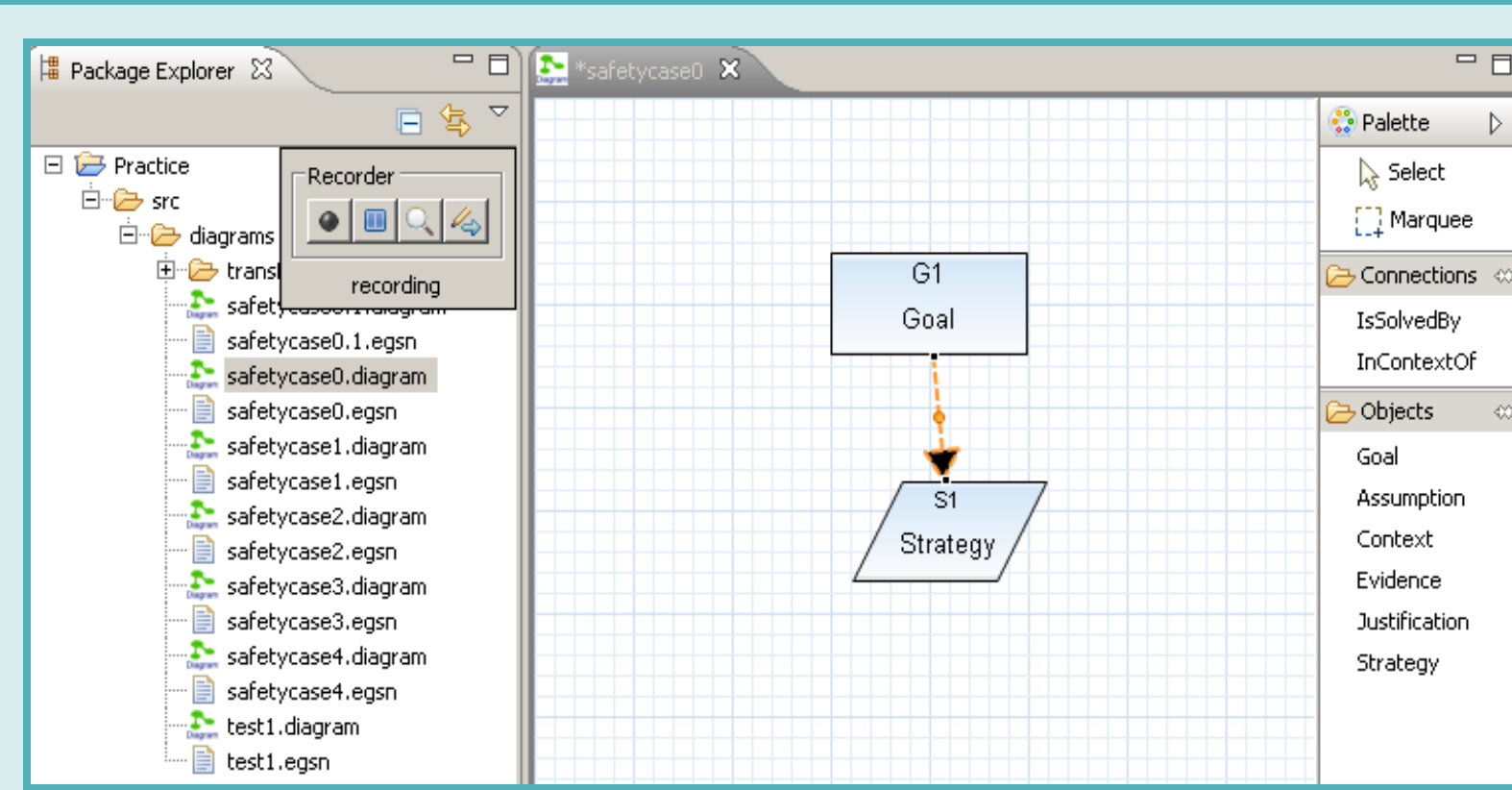
Figure 2. A sample Safety Case Spreadsheet of the Fig 1. diagram

Testing the Safety Case Editor

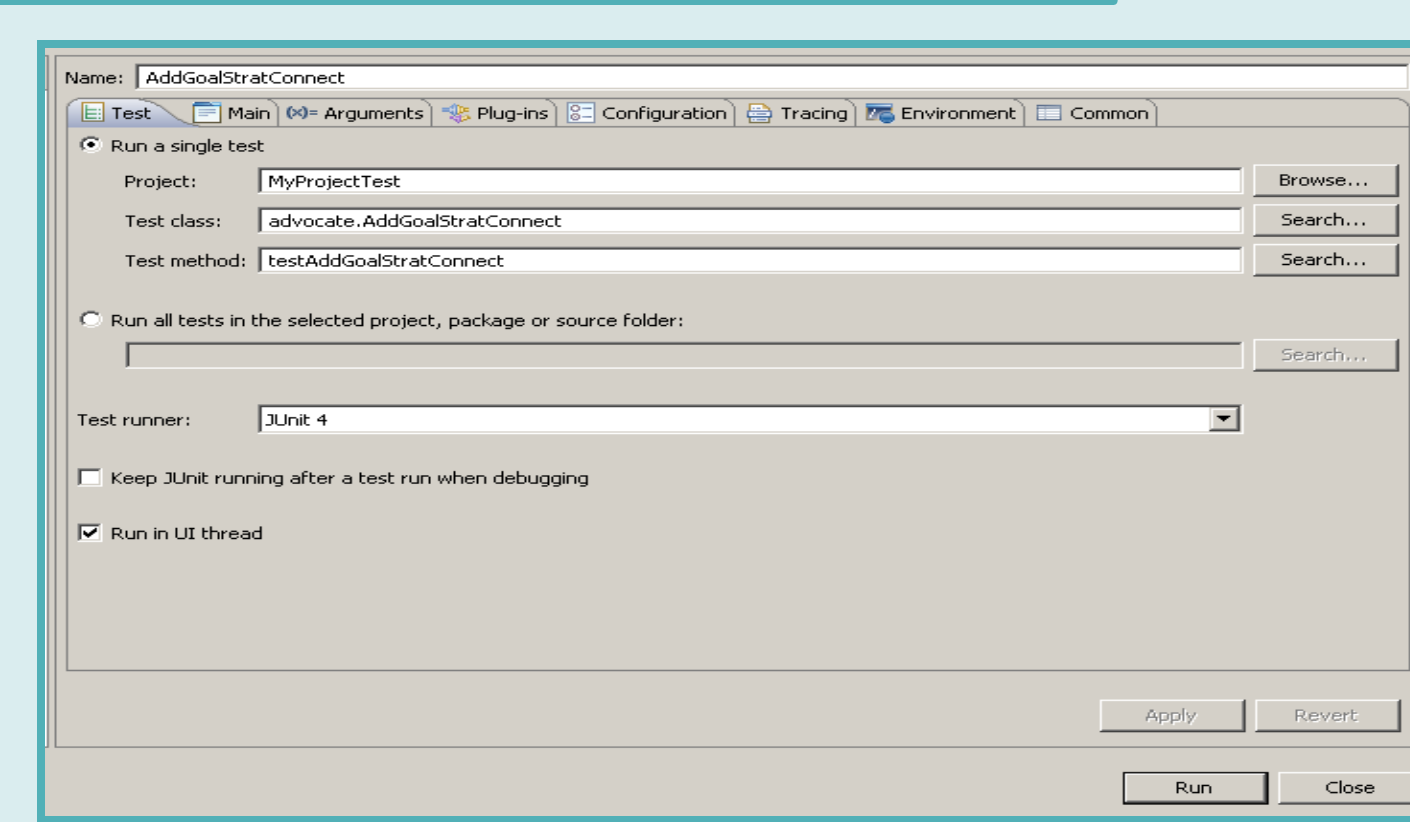
After changes have been made to the safety case editor, it's necessary to test the editor to make sure the changes have not negatively affected its functionality.



Testing the functionality of adding nodes and links using WindowTester Pro:



STEP 1: Record all keyboard clicks, mouse movements, and all changes made to the diagram, in this case a Goal and Strategy were added and then an IsSolvedBy link was added.



STEP 2: Run the newly created junit plug-in test and view the results of the test. Edit the produced java code if necessary to make it run correctly.