Submitted to the 2nd International Workshop on Living With Inconsistency
co-located with the 23rd International Conference on Software Engineering (ICSE 2001), Toronto, Canada
http://www.cs.toronto.edu/˜sme/IWLWI-01/
WP ref: 01/conflict/conflict (March 15, 2001).

Funnelling the Consequence of Inconsistencies

Tim Menzies
Dept. Electrical Engineering & Computer Engineering, University of British Columbia, Vancouver, Canada;
<tim@menzies.com>

March 15, 2001

Unresolved conflicts can introduce inconsistent and uncertain choices into a system description. Such inconsistencies imply nondeterminancy and nondeterminism is a *bad thing*; e.g. Leveson says that "nondeterminism is the enemy of reliability" [2].

Requirement engineers, on the other hand, argue that inconsistencies are a *good thing* [9]. To be human is to hold an opinion. To be an expert is to hold an opinion that others wish to pay for. Hence, a room of experts must argue, lest any one of them loses their income. Inconsistencies are not embarrassments that should be left undocumented. Rather, their detection, exploration and partial resolution are a powerful propellant to drive option discovery and documentation. Contrary to Leveson's views, inconsistencies can make a system *safer* since unsafe systems typically result from an unexpected consequences. Exploring inconsistencies can drive a design into a zone that was not previously considered where unsafe possibilities can be recognized and repaired.

But the arguments of requirements engineers may not convince the broader software engineering community. If we can't demonstrate that some stable set of consequences can be inferred from a space of inconsistent assertions, then our requirements will appear unpredictable and untrustworthy. Theoretically, such a demonstration is intractable. Gabow et.al. [1] showed that building pathways across programs with inconsistent pairs (e.g. $x\ and\ \neg x$) is *NP-hard* for all but the simplest software models (a software model is very simple if it is very small, or it is a simple tree, or it has a dependency networks with out-degree ≤ 1). No fast and complete algorithm for NP-hard tasks has been discovered, despite decades of research. Hence, computing all the consequences from a space of inconsistencies can be impossibly slow, except for very small models.

Empirical results offers new hope for the practicality of exploring a space of inconsistent choices. Menzies, Easterbrook, Nuseibeh and Waugh [8] found that most of the choices made within a space of conflicts had the same net effect. That study compared two search strategies. In *full worlds search*, one *world of belief* was forked for each possible resolution to some inconsistency. In *random worlds search*, when N worlds are possible, one was picked at random. In a very large case study (over a million runs), Menzies, Easterbrook, Nuseibeh and Waugh found that the average difference in reachable goals between the random worlds search and full worlds search was less than 6% (!!).

These results can be explained via the *funnel theory* first proposed by Menzies, Easterbrook, Nuseibeh and Waugh [8], then elaborated by Menzies, Singh, Powell, and Kiper [4–7]. To introduce funnels, we first say that an argument space supports *reasons*; i.e. chains of reasoning that link inputs in a certain context to desired goals. Chains have links of at least two types. Firstly, there are links that clash with other links. Secondly, there are the links that depend on other links. For example, suppose the following argument space is explored using the invariant $nogood(X, \neg X)$ and everything that is not a $context$ or a $goal$ is open to debate:

$$\begin{array}{c}
a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow e \\
context1 \longrightarrow f \longrightarrow g \longrightarrow h \longrightarrow i \longrightarrow j \ \ \longrightarrow goal \\
context2 \longrightarrow k \rightarrow \neg g \longrightarrow l \longrightarrow m \rightarrow \neg j \ \longrightarrow goal \\
n \longrightarrow o \longrightarrow p \longrightarrow q \longrightarrow \neg e
\end{array}$$

While all of $\{a, b, ..q\}$ is subject to discussion, in the context of reaching some specified goals from *context1* and *context2*, the only important disputes are the clashes $\{g, \neg g, j, \neg j\}$. The $\{e, \neg e\}$ clash is not exercised in the context of $context1, context2\ \vdash\ goal$ since no reason uses e or $\neg e$. Since $\{j, \neg j\}$ are fully dependent on $\{g, \neg g\}$, then the core of this argument is one variable $\{g\}$ with two disputed values: true and false.

The *funnel* of an argument space contains the non-dependent clashing links; in this case, $\{g\}$. The arguments with *greatest information content* are the arguments about the funnel variables, since these variables set the others. Suppose our stakeholders agree that g is true, then in the context of arguing about how $context1, context2\ \vdash\ goal$, the argument space reduces to:

$$context1 \longrightarrow f \longrightarrow g \longrightarrow h \longrightarrow i \longrightarrow j \longrightarrow goal$$

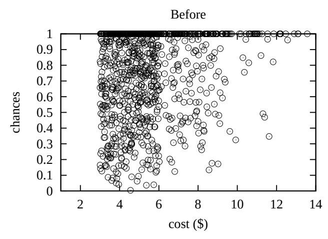

Figure 1.A: $Cost$ (x-axis) and $chances$ (y-axis) from 1,000 stochastic runs.

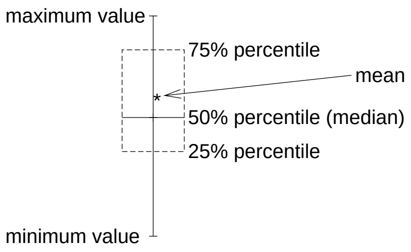

Figure 1.B: Box plots: explanation of symbols.

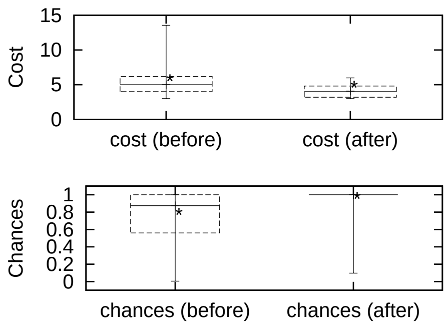

Figure 1.C: Changes in the $costs$, and $chances$ before and after "funnel-Var=false".

Figure 1: Experiments with stochastic search

The reasoning starting with k has been culled since, by endorsing g, we must rejects all lines of reasoning that use $\neg g$. Also, the reasoning starting with a, n are ignored since they are irrelevant in this context; i.e. they do not participate in reaching a desired goal. Further, in this context, there is little point arguing about $\{f, h, i, j\}$ since if any of these are false, then no goal can be reached.

Funnels reduce the number of consequences that can be inferred from a space of inconsistencies. The above example, showed a space of up to $2^{16} = 65536$ discussions about 16 boolean variables $\{a..q\}$. Funnels reduced this space to one issue that controlled all the other consequences: i.e. "is g true or false?". Funnel theory explains the above empirical results. Most of the choices explored by Menzies, Easterbrook, Nuseibeh, Waugh had the same net effect since most of them were not within the funnel.

Requirement engineers can exploit funnels using a technique called *stochastic search*. Any randomly selected pathway from inputs to goals must pass through the funnel (by definition). Hence, after a limited random sampling of a space of options, (i) part of the range of the funnel variables must have been reached; (ii) a few variables can be detected that significantly constrain the behaviour of the whole system; and (iii) the range in the consequences of the inconsistencies can be greatly reduced. In one study of this technique, Menzies and Kiper ran 1000 stochastic searches over a rule base containing heuristic costs and likelihoods. The rule base contained many contradictions which the inference engine resolved by a random selection of a resolution. Further, the costs and likelihoods were also uncertain and were varied over a wide range prior to each run. The results in Figure 1.A show a large spread in the final costs and likelihoods (chances). This result is intuitively obvious: given inconsistencies and unknowns, a wide range of behaviour can be generated. However, because of funnels, the range of this behaviour can be greatly reduced. Using some simple induction techniques, Menzies and Kiper found a funnel containing one attribute range that significantly constrained the rest of the system. The box-plots of Figure 1.C show how setting that funnel variable to "false" significantly constrained the consequences (note the reduction in the range of both the cost and chances of achieving the system's goal).

Stochastic search algorithms are simple to build or download from the web. Elsewhere, it has been argued that, in the usual case, funnels are common and they greatly constrain the space of all possible inferences [3]. Hence, under the assumptions of funnels and stochastic search, the consequences of inconsistencies can be constrained, and we need not view inconsistencies as a *bad thing*. This is a *good thing*.

References

[1] H.N. Gabow, S.N. Maheshwari, and L. Osterweil. On two problems in the generation of program test paths. *IEEE Trans. Software Engrg*, SE-2:227–231, 1976.

[2] N. Leveson. *Safeware System Safety And Computers*. Addison-Wesley, 1995.

[3] T. Menzies and B. Cukic. Adequacy of limited testing for knowledge based systems. *International Journal on Artificial Intelligence Tools (IJAIT)*, June 2000. To appear. Available from `http://tim.menzies.com/pdf/00ijait.pdf`.

[4] T. Menzies and J.D. Kiper. How to argue less. In *Submitted to RE01*, 2001. Available from `http://tim.menzies.com/pdf/01jane.pdf`.

[5] T. Menzies and H. Singh. How AI can help SE; or: Randomized search not considered harmful. In *AI'2001: the Fourteenth Canadian Conference on Artificial Intelligence, June 7-9, Ottawa, Canada*, 2001. Available from `http://tim.menzies.com/pdf/00funnel.pdf`.

[6] T. Menzies and H. Singh. Many maybes mean (mostly) the same thing. In *2nd International Workshop on Soft Computing applied to Software Engineering (Netherlands), February*, 2001. Available from `http://tim.menzies.com/pdf/00maybe.pdf`.

[7] Tim Menzies, Bojan Cukic, Harhsinder Singh, and John Powell. Testing nondeterminate systems. In *ISSRE 2000*, 2000. Available from `http://tim.menzies.com/pdf/00issre.pdf`.

[8] T.J. Menzies, S. Easterbrook, Bashar Nuseibeh, and Sam Waugh. An empirical investigation of multiple viewpoint reasoning in requirements engineering. In *RE '99*, 1999. Available from `http://tim.menzies.com/pdf/99re.pdf`.

[9] B. Nuseibeh, S. Easterbrook, and A. Russo. Leveraging inconsistency in software developoment. *IEEE Computer*, 33(4):24–29, April 2000. Available from `http://www-dse.doc.ic.ac.uk/˜ban/pubs/computer2000.pdf`.