Background, Related Literature, Objectives: Much of modern research into software engineering (SE) is out of step with much of current industrial practice. At the recent International Symposium on Software Reliability (San Jose, California, 2000), a keynote address from Sun Microsystems shocked the researchers in the audience: few of the techniques endorsed by the SE research community are being used in fast-moving dot-com software companies. For such projects, developers and managers lack the resources to conduct *heavyweight software modeling*; e.g. the construction of complete descriptions of the business model or the user requirements. Yet such heavyweight software modeling is very useful. Complete models of (e.g.) specifications can be used for useful tasks such as:

- Auto-generation of test cases from the specification;
- Finding redundant or unused parts of a software model;
- Understanding the consequences of conflicts between the requirements of different stakeholders;
- Testing that temporal constraints hold over the lifetime of the execution of the specification.

To better support the fast pace of modern software, we need a new generation of *lightweight software modeling* tools. Lightweight software models can be built in a hurry and so are more suitable for the fast-moving software companies. However, software models built in a hurry can contain incomplete and contradictory knowledge. The presence of contradictions in the lightweight theories complicates the above useful tasks. Suppose some inference engine is trying to build a proof tree across a lightweight software model containing contradictions. Gabow et.al. [7] showed that building pathways across programs with contradictions is *NP-complete* 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-complete tasks has been discovered, despite decades of research.

Empirical results offers new hope for the practicality of NP-complete tasks such as reasoning about lightweight software models. A repeated and robust empirical result from the satisfiability community (e.g. [1, 20]) is that theoretically slow NP-complete tasks are only truly slow in a narrow *phase transition* zone between under-constrained and over-constrained problems. Further, it has been shown empirically that in both the under/over-constrained zones, seemingly naive randomized search algorithms execute faster than, and nearly as completely, as traditional, slower, complete algorithms [20]. It is easy to see why this might be so. In the over-constrained zones, it is impossible to satisfy all constraints so we need not search very long. In the under-constrained zone, many solutions exist so, once again, we need not search very long.

These empirical results suggest that we might be able to simply the processing of lightweight software models using randomized search. This project will determine if and when we can:

- *Expect* randomized search over lightweight software models to be fast and effective (short-term goal).
- *Exploit* the fast runtimes of that randomized search (long-term goal).

Note that what we can't do is simply translate the results from the satisfiability community into SE. The predictors for the phase transition zone are expressed in a form that is too low-level for the average software engineer; e.g. Selman's linear clause model does not refer to design structures that the average software engineer would recognize [20]. Recent results suggest that the available predictors for the phase transition zone are incomplete [10]. We have some preliminary results strongly suggesting that we can get better and more detailed predictors for the phase transition zone by assuming theories are expressed as *horn clauses* and not the conjunctive normal form using by the satisfiability community [15][1]. For example, these results

[1]This paper is attached to this application. See *Testing Nondeterminate Systems* by Menzies et.al., ISSRE 2000.

can compute a mathematical probability that randomized search will be an adequate search strategy for particular systems. Computing this probability will be essential if randomized search is to be applied to safety-critical software.

This research hence focuses on *random abductive search over horn clause theories*. In many domains, software engineers can understand *horn clause* representations of their models (for example, database modelers find it easy to map from SQL into the horn clauses of Prolog). *Abduction* is a natural method of processing theories containing contradictions. When abduction finds a contradiction, it forks one world of belief for each possibility. Each world *fixes* the uncertainties in a theory by committing to a particular set of consistent assumptions [12]. *Randomized abduction* explores a small number of randomly selected fixes.
Significance: If we can't show that dependable conclusions can be drawn from lightweight software models, then we cannot use them widely. Nor can we adapt heavyweight SE principles to the needs of fast-paced software companies.

If we can predict where fast random search will suffice for lightweight software modeling, then we could better support the fast-pace of modern SE. For example:

- Suppose we could *define lightweight design principles* that always lead to software models that can be quickly and adequately probed via randomized search. For those systems, we can quickly discover the implications of the uncertainties within our lightweight software models.
- Also, we could use our *randomized abductive theorem provers* to optimize tasks such as generating test cases from specifications, diagnosing the cause of faulty outputs, or understanding the consequences of conflicting requirements from different stakeholders.
- Further, we could *define early stopping rules* for the testing a specification. In systems where fast random probing will suffice, a small number of random probes will reveal most of what we will reach via a much larger number of probes.

Related Recent Progress: Since 1995, I have been studying randomized abduction over horn clauses and their application to SE and knowledge engineering. For example, one study compared two abductive inference engines for analysis inconsistent requirements models. In that study, a complete abductive procedure explored all consequences while the other procedure just build a small number of randomly selected worlds. The random search found nearly all the goals found by the complete search [16]. This result can be explained via a narrow phase transition zone within horn clauses: if most of the inference occurred outside the transition zone, then a random inference procedure should perform as well as a rigorous search. Nevertheless, this study might just be some quirk of the models being studied. Hence, several further studies were conducted.

- A literature review by Menzies and Cukic found numerous cases in the SE and knowledge engineering literature where a small number of random probes into a system yielded as much information as a much larger number of probes [13].
- Menzies et.al. developed a general mathematical model of random abductive search which, on simulation, found that a small number of random probes into a system usually yielded as much information as a much larger number of probes [14, 15].

The results of these studies are clearly consistent with the widespread nature of the phase transition effect. Further, these studies suggest that we can expect randomized abduction to terminate quickly in a wide range of applications. These results prompted the development of HT0 [17], a new randomized abductive inference engine. HT0 generates a few randomly selected worlds and terminates when the best coverage of the goals seen so far stops increasing. When compared to a complete abductive inference procedure, HT0 runs much faster, runs on much larger theories, and finds nearly as many goals as the complete search.

Methods and Proposed Research: The *short-term goal* is to build a good predictor for when we can expect the phase-transition effect within lightweight software models expressed as horn clauses. This requires two projects:

- *Project 1: A theoretical analysis* The Menzies et.al. mathematical model of abductive search makes certain predictions about what kinds of horn clauses will/won't be probed by fast random search [15]. Project one would test those predictions via numerous runs over artificially generated horn clauses. Part of this project will be to update the mathematical model if the runs reveal inaccuracies in the predictions.

- *Project 2: Case studies:* This researcher can access a range of real-world theories including:
  - Software time and risk estimation models (e.g. COCOMO-II);
  - Models of the requirements of particular systems from NASA's Jet Propulsion Laboratory;
  - State charts describing procedural programs generated from the finite state machines created by the BANDERA program slicing tool [3] or the SPIN tool [9].

  Using a combination of manual modeling and automatic translation techniques, these theories can be expressed as horn clauses. Once expressed as horn-clauses, we can search them using complete and randomized abduction:
  - If the predictor built in the theoretical analysis is any good, it should be able to predict how well random search will probe these real-world theories.
  - If the phase transition effect is as widespread as we suspect, then we would predict that randomized search would adequately probe these models very quickly.

The *long term goal* of this project is to exploit theories which we expect to be suitable for random probing. To test if we can exploit the phase transition effect, we need to undertake projects 3&4:

- *Project 3: Applications work:* For the real-world theories described above, we need to build test case generators, diagnosis devices, and define early stopping rules for testing (e.g. we can stop testing early when our phase transition predictor tells us that this theory needs only limited random probing).

- *Project 4: Guidance work:* We seek design principles for lightweight software modeling that lead to the creation of lightweight software models that can be probed via randomized abduction.

Success Criteria:

Project	Success criteria
1:	For artificially generated sets of horn clauses: • We can predict from the mathematical theory what percentage of goal literals can be reached from N random inputs. • We can predict from the mathematical theory when nondeterminism will/won't imply that reachable conclusions are/aren't *stable*. *Stability* means that nondeterminism in a model does not mean that N random runs will generate many different conclusions.
2:	For horn clauses generated from real-world models, we can make the same predictions as for artificially generated sets of horn clauses (see Project 1's success criteria).
3:	Test case generation, diagnosis via randomized search terminates in quadratic (not exponential) time with satisfactory and *stable* results.
4:	We can conduct case studies where developers do/don't use our guidelines and the systems built using our guidelines are much more testable that those developed without.

Many of the applications described in the long term goals have been explored before in the literature. However, the claim to be tested here is that we can effectively and simply perform these tasks using randomized abduction over lightweight descriptions of systems that can contain contradictions. Further, these tasks should take quadratic time, not exponential time, to execute.

More generally, this project will succeed if:

- If non-trivial horn clauses can be adequately probed by randomized abduction. Our preliminary results suggest that this is not the case, but this should be monitored within this project.
- If horn clauses generated from our SE case studies can be adequately probed by randomized abduction. Our case study work will reveal if this is so.
- Also, we will have a principled method for developing the guidance work described above.

Related Literature: The general problem of reasoning about inconsistent SE models has been well studied. For example, in the SE literature, requirements engineering researchers have explored conflicting requirements generated either from non-functional requirements (e.g. [18]) or from multiple stakeholders (e.g. [5, 6]). A standard technique for this exploration is some contradiction tolerant logic such as the LTMS [18] or the quasi-consistent logics [11]. This research is an attempt to simplify the standard technique. In systems where random search can adequately probe a space of uncertainties, then very simple inference techniques will suffice to probe the space of "maybes". If we can demonstrate that randomized search is widely useful, then we can design a new generation of very simple contradiction tolerant reasoners.

Abduction has also be extensively studied and applied to fields such as model-based diagnosis [8, 12]. Abduction is the logic of argument. An abductive proof must not only reach goals, but it must also enforce consistency within that proof. When conflicts arise in the reasoning, abduction generates worlds of belief to store the conflicting possibilities. Abduction is impractical when too many worlds are generated. For example, consistency-based abduction (e.g. the ATMS [4]) adds all literals consistent with inputs and goals are added to it's worlds. This can very slow; e.g. all known ATMS implementations have runtimes exponential on theory size [19]. Hence this research uses set-covering abduction instead of consistency-based abduction [2]. In set-covering abduction, literals are only added to worlds if those literals appear on proof trees that link inputs to goals. This strategy can be faster than consistency-based reasoning but is still NP-complete. This research hopes to tame the runtimes of NP-complete abduction using random search.

For notes on other related work, in particular the phase transition research of the satisfiability community, see the earlier pages of this document.

Training: This work would generate three masters and three Ph.D. students trained in mapping abstract logic programming to real world SE problems.

These students would be assigned to the four projects listed in the *Methods and Proposed Research* section (and the projects within the short term goals would be addressed before the projects within the long term goals).

References

[1] P. Cheeseman, B. Kanefsky, and W. Taylor. Where the really hard problems are. In *Proceedings of IJCAI-91*, pages 331–337, 1991.

[2] L. Console and P. Torasso. A Spectrum of Definitions of Model-Based Diagnosis. *Computational Intelligence*, 7:133–141, 3 1991.

[3] J. Corbett, M. Dwyer, J. Hatcliff, S. Laubach, C. Pasarenu, Robby, and H. Zheng. Bandera: Extracting finite-state models from java source code. In *Proceedings ICSE2000, Limerick, Ireland*, pages 439–448, 2000.

[4] J. DeKleer. An Assumption-Based TMS. *Artificial Intelligence*, 28:163–196, 1986.
[5] S. Easterbrook. Handling conflicts between domain descriptions with computer-supported negotiation. *Knowledge Acquisition*, 3:255–289, 1991.
[6] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handling in multi-perspective specification. *IEEE Transactions on Software Engineering*, 20(8):569–578, 1994.
[7] H. Gabow, S. Maheshwari, and L. Osterweil. On two problems in the generation of program test paths. *IEEE Trans. Software Engrg*, SE-2:227–231, 1976.
[8] W. Hamscher, L. Console, and J. DeKleer. *Readings in Model-Based Diagnosis*. Morgan Kaufmann, 1992.
[9] G. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997.
[10] H. Hoos and C. Boutilier. Solving combinatorial auctions using stochastic local search. In *Proc. of AAAI-2000*, pages 22–29. MIT Press, 2000.
[11] A. Hunter and B. Nuseibeh. Analysing inconsistent specifications. In *International Symposium on Requirements Engineering*, pages 78–86, 1997.
[12] A. Kakas, R. Kowalski, and F. Toni. The role of abduction in logic programming. In C. H. D.M. Gabbay and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming 5*, pages 235–324. Oxford University Press, 1998.
[13] 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`.
[14] T. Menzies and B. Cukic. When to test less. *IEEE Software*, 17(5):107–112, 2000. Available from `http://tim.menzies.com/pdf/00iesoft.pdf`.
[15] T. Menzies, B. Cukic, H. Singh, and J. Powell. Testing nondeterminate systems. In *ISSRE 2000*, 2000. Available from `http://tim.menzies.com/pdf/00issre.pdf`.
[16] T. Menzies, S. Easterbrook, B. Nuseibeh, and S. Waugh. An empirical investigation of multiple viewpoint reasoning in requirements engineering. In *RE '99*, 1999. Available from `http://tim.menzies.com/pdf/99re.pdf`.
[17] T. Menzies and C. Michael. Fewer slices of pie: Optimising mutation testing via abduction. In *SEKE '99, June 17-19, Kaiserslautern, Germany. Available from `http://tim.menzies.com/pdf/99seke.pdf`*, 1999.
[18] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions of Software Engineering*, 18(6):483–497, June 1992.
[19] B. Selman and H. Levesque. Abductive and Default Reasoning: a Computational Core. In *AAAI '90*, pages 343–348, 1990.
[20] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *AAAI '92*, pages 440–446, 1992.