Uncertainty as the Source of Knowledge Transfer Opportunity

Alex Dekhtyar, Jane Hayes and Judy Goldsmith

ABSTRACT

Uncertainty creeps into the software development process in many ways, shapes and forms. In the early stages of software development, key sources of uncertainty are the human stakeholders who help formulate the requirements of the software product. An added layer of uncertainty is inherent as requirements analysts have to deal with subjective, and often conflicting, estimates that humans make, estimates that may significantly affect both the software development process and the eventual software product. Our position is two-fold. We stipulate that in situations where analysts (and later developers) have to deal with human evaluations of uncertainty, special methods and procedures should be used to elicit this information, reconcile this information, and, most importantly, use this information for decision-making. We also note that significant developments are unfolding in the field of Artificial Intelligence in two areas related to dealing with uncertainty: eliciting data from domain experts, and using uncertain data for inference and planning. We believe that mitigation (and proper use) of uncertainty in the early stages of software development calls for collaboration between the fields of Software Engineering and Artificial Intelligence.

1. INTRODUCTION

Uncertainty exists (and often persists) in virtually every human endeavor. Software development is no exception. On the one hand, psychological research of the past 50 years [3,15,16] suggests that humans are notoriously poor and inconsistent estimators of uncertainties associated with their activities. But there are, at times, severe penalties associated with a human's *inability* to correctly estimate the certainty of events, draw correct inferences, and construct long-term plans that take uncertainty into account.

The software engineering process, regardless of lifecycle, programming language, or domain, involves significant human decision-making. Often, the process of building software starts with a collaboration of parties with expertise in completely different areas: software engineers, requirements analysts, domain experts, customers, end users, knowledge engineers, etc. A software engineer may know how to construct specifications and build software, while the customer may have zero knowledge in that area. But the customer may know their domain very well and may know what type of software is needed. In such situations, uncertainty stems from a number of sources, including:

- misunderstandings between software engineers and domain experts;
- shortcomings in or inability to estimate the parameters of post-deployment software operation in advance;
- conflicting opinions of domain experts; and
- conflicting priorities of stakeholders such as managers, and workers, etc.

Uncertainty exists both in the software development process (How long will it take? How much will it cost? What is the most efficient process to develop this software? What is the right choice of programming environment/language?) and in the software itself (How much security, critical error-handling, and recovery needs to be implemented? What is the right trade-off between efficiency and security, or between efficiency and functionality? What features of the software will be used most heavily?). Probability theory, as well as other formalisms for dealing with uncertain information (fuzzy logic/set theory, belief theory), provide guidelines for handling

uncertain information, once uncertainties have been established and properly quantified. However, in many applications involving human participation, the process of establishing uncertainties and quantifying them presents the greatest challenge [13,14,15,16]. When we start a software project, we may be able to identify exactly where uncertainty appears (cost, time, workloads, etc.). We must rely, however, on subjective human opinion (and experts who may not agree on anything) to estimate these uncertainties. In certain situations, even discovering *what* is uncertain is a challenge. When a brand new type of software is built, the lack of computer-related expertise of the customer may prevent the customer from even realizing what is feasible and what is not.

The good news is that these issues are not unique to software development. Elicitation of uncertain information and quantification of qualitative uncertainties are topics currently receiving much attention from the artificial intelligence researchers [18,13,8,9], as well as psychologists [3,16,17], economists, and decision scientists [15]. In the rest of this paper, we briefly discuss two things: the software development scenarios that lead to human-generated uncertainties and the possibilities for applying artificial intelligence (AI) research to such scenarios.

2. UNCERTAINTY IN EARLY STAGES OF SOFTWARE DEVELOPMENT

We posit that there are three research areas of importance when handling human evaluations of uncertainty: elicitation of uncertainty information, reconciliation of such information, and application of such information to decision-making. Further, we posit that AI research can provide techniques for the elicitation of uncertainty information from domain experts and for using uncertain data for inference and planning.

To illustrate our ideas, we present two scenarios to show how uncertainty might appear in the software engineering process and/or product. We also suggest ways how existing AI work might assist in dealing with the uncertainty.

Scenario 1. Uncertainty in final product.

Consider the following software development scenario. A team of developers is contracted to develop and deploy software for a portable communications computer used by NASA on manned space missions. A small communications computer is strapped on the astronaut's space suit.

The software is responsible for the quality of the communications, as well as their reliability and robustness. The software can be designed to be very robust, running several processes in parallel in case one fails, so that there is never a loss of the communications lines. However, this comes at a penalty of a delay in the transmission and reception of messages. Based on this knowledge, the requirements engineer begins to solicit requirements from the appropriate NASA stakeholders.

Three stakeholders represent NASA in this process: the NASA mission control coordinator, the space mission commander, and the mission specialist: the astronaut who will eventually be the end-user of the software and the hardware. The mission coordinator wants to maximize the reliability and would prefer to run three or four processes in parallel to ensure that communications are never lost. Based on his experience with previous missions, he estimates that there is a high probability that a single-process system will result in the loss of important communications. The mission commander feels that getting information after a delay makes that information almost useless and estimates that the chance of missing a very important

communication is small enough that it does not outweigh the need for immediacy in communication. The mission specialist has a middle-of-the road view. She thinks that in the long term, the probability of deteriorated communications is high enough to warrant some level of redundancy, but she also thinks that any significant transmission delay would hamper her work during the spacewalk and, thus, wants a compromise solution.

In this scenario, the development team has to decide on the eventual set of requirements, and address directly the key controversy: how much redundancy is going to be implemented, and how the redundancy mechanism will operate in the software. The key challenge lies in the fact that the expert assessments of failure chances and risks associated with communications failure are different, possibly even contradictory. The analysis of the situation turns into a two-step process.

1. Reconcile the conflicts in the assessment of failure chances between experts, derive final assessments.
2. Use obtained assessments in a risk-analysis procedure (e.g., probabilistic inference) to determine the desired level of redundancy.

Scenario 2. Uncertainty about process.

A software development team receives a project which comes with a predefined budget, deadlines, and a well-defined overall task. The project manager has to work within given parameters and needs to find the best way to task different members of the team. She knows the strengths and weaknesses of the team members. Some are better at formal tasks, such as requirements engineering, some are better designers, and some are better developers. Some work faster, but their code may be error-prone. Some produce code that passes all tests, but takes more time to deliver it to the testers. Some, like the abovementioned testers, have a very limited scope of responsibility. In the absence of work, these team members represent a loss of person-hours devoted to the project. What is the best way for the project manager to proceed?

In this scenario, the uncertainty is associated with the selection of the correct process for software development. If the project manager were willing to express her knowledge about the quality and speed of the employees' work in terms of probabilities (e.g., "*there is about a 75% chance that Steve will finish implementing this functionality within 3 weeks" or "there is an 80% chance that Mary's code will contain no severe defects", "If Mary's code contains no severe defects, Paul can finish testing it within one week with probability 85%")*), and if she were willing to quantify the *utility* (i.e., the expected benefit of assigning different tasks to different people), then stochastic planning mechanisms[1,2], such as planning with Markov Decision Processes [13,5], can be used to obtain a policy. Policies are functions from states to actions. From a policy, the project manager will obtain not just a single sequence of suggested assignments (*Mary elicits requirements, Paul does design, Steve programs GUI, Mary programs backend, Paul tests*), but will receive suggestions that depend on the results of previous actions (*Mary elicits requirements; if she does it in three weeks, Paul does design, otherwise, Steve does it*). A policy will cover all eventualities.

With these scenarios in mind, we move to a discussion of what can be done to address these types of uncertainty.

3. WHAT CAN BE DONE

Two issues arise from the scenarios in the previous section. We address each in turn.

Issue 1. Elicitation of Uncertain Information.

In each scenario (and in many other scenarios involving humans), the first step toward the solution is obtaining reasonable, trustworthy, and consistent estimates of the certainty/uncertainty of events. In the examples above, we suggested elicitation of probabilistic estimates about the events. Other measures of uncertainty (fuzzy measures, measures of belief) could be elicited, depending on the nature of uncertainty in the problem and the expertise of programmers; our expertise is in probabilistic methods.

Uncertainty elicitation can be broken into two sub-problems.

1. Elicitation of probabilities from individual experts, and
2. consolidation of elicited probabilities, including resolution of conflicts.

Research has shown that different categories of people have different levels of comfort with the measures of uncertainty. Development of good procedures for eliciting probabilities (and in some cases, eliciting *many* probabilities) is an emerging topic in AI research. Various methods [14,18,9] have been proposed and studied on different populations of users. There has not been, to our knowledge, any research involving similar studies with domain experts representing customers of large software projects.

Similarly, resolution of conflicts in probability assessments has been a topic of study in recent years. Several complimentary methods were proposed recently [6,11,12]. Some methods rely on statistical procedures which determine the most likely probabilities/ranges [6]. Some methods use extra information to assign "weights" to the opinions of each expert and compute the final assessment as the weighted average [11]. Some methods represent uncertainty in terms of probability intervals or other imprecise measures [4].

Issue 2. Use of Elicited Uncertain Information.

In the scenarios discussed in Section 2, we mentioned two different types of procedures that could use the probabilistic assessments. In the first scenario, risk assessment could be conducted in a reasonably straightforward manner by combining the probabilities of events with the utilities of various situations. Bayesian networks [19] and Bayesian inference can be used, if probability assessments are conditional (*e.g., the probability of communications failure depends on: (a) the charge in the communications computer battery, (b) solar activity, and (c) the amount of time the computer spends in vacuum. The latter depends on (d) the nature of work (space walk) and (e) the astronaut completing the walk.).*

In other situations, such as the one described in the second scenario, simple inference is not enough. The goal of the project manager is not to predict the most probable state (e.g., the most probable way in which the software will be used), but rather to *plan* [2,7] what actions to take to achieve a goal or acquiring utility, in an ongoing scenario, or both. As mentioned above, a policy computed by a program that handles uncertainty (such as SPUDD [5]) provides advice for every possible state of the product development process in the foreseeable future.

4. THE CHALLENGE TO THE SOFTWARE ENGINEERING COMMUNITY

Software engineers are not alone in their attempts to deal with uncertainty, but the ball is in their court in terms of taking actions. While the problems of reasoning with uncertainty are often computationally hard [7,10] in general cases, many useful heuristics have been suggested and implemented that make inference and planning feasible for reasonably large problems. What remains to be seen is how this can be used by software engineers and incorporated into the software development process. We suggest that empirical studies be undertaken for determining

a. the best means of eliciting probabilities from customers of software development projects (or domain experts), and
b. the applicability and the cost benefit of using probabilistic reasoning in software development projects.

These studies should be undertaken by software engineering and artificial intelligence researchers working in concert. Together, the researchers can address the important problem of dealing with uncertainty in software process and software products.

5. ACKNOWLEDGMENTS

Our work is partially funded by NASA under grant NAG5-11732 and by the National Science Foundation under grant ITR-0325063.

6. REFERENCES

[1] Jim Blythe. 1999. Decision-Theoretic Planning, *AI Magazine*, Vol. 20, No. 2, pp. 37-54.

[2] Craig Boutilier, Thomas Dean, Steve Hanks. 1999. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage, *Journal of AI Research*, Vol. 11, pp. 1-94.

[3] L. Cosmides and J. Tooby. 1996. Are humans good intuitive statisticians after all: Rethinking some conclusions from the literature on judgment under uncertainty. *Cognition*, Vol. 58, pp. 1-73

[4] Luis M. de Campos, Juan F. Huete, Serafín Moral. 1994. Uncertainty Management Using Probability Intervals, in *Proc. IPMU 1994*, pp. 190-199.

[5] Jesse Hoey, Robert St-Aubin, Alan Hu, Craig Boutilier. 1999. SPUDD: Stochastic Planning using Decision Diagrams, in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pp. 279-288.

[6] Gabriele Kern-Isberner, Wilhelm Rödder. 2004. Belief revision and information fusion on optimum entropy. *Int. J. Intell. Syst.* Vol. 19(9), pp. 837-857

[7] M. Littman, J. Goldsomith, M. Mundhenk. 1998. The Computational Complexity of Probabilistic Plan Existence and Evaluation, *The Journal of AI Research*, Volume 9, pages 1--36, 1998.

[8] Krol Kevin Mathias, Casey Lengacher, Derek Williams, Austin Cornett, Alex Dekhtyar, Judy Goldsmith. 2006. Factored MDP Elicitation and Plan Display, demo, in *Proc. AAAI'06 Conference*, July 2006, Boston, MA.

[9] Krol Kevin Mathias, Cynthia Isenhour, Alex Dekhtyar, Judy Goldsmith, Beth Goldstein. (2006). When Domains Require Modeling Adaptations, in *Proc., 4th Bayesian Modelling Applications Workshop at UAI06*, July 2006, Boston, MA.

[10] Martin Mundhenk, Judy Goldsmith, Christopher Lusena and Eric Allender. 2000. Complexity Results for Finite-horizon Markov Decision Process Problems, *Journal of the ACM*, Vol. 47. No. 4, pp. 681-720.

[11] Pedrito Maynard-Zhang, Daniel J. Lehmann. 2003. Representing and Aggregating Conflicting Beliefs. *J. Artif. Intell. Res. (JAIR)*, Vol. 19, pp. 155-203.

[12] Stefano Monti and Giuseppe Carenini. 2000. Dealing with the expert inconsistency in probability elicitation, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 4, pp. 499—508.

[13] Matin L. Puterman. 1994. *Markov Decision Processes*, John Wiley & Sons, New York.

[14] Silja Renooij, Cilia Witteman. 1999. Talking probabilitites: communicating probabilistic

information with words and numbers, *International Journal of Approximate Reasoning*, Vol 22, No. 3, pp. 195-215.

[15] C. Spetzler, C-A. Staël von Hostein. 1975. Probability encoding in decision analysis, *Management Science*, Vol. 22, pp. 340-358.

[16] A. Tversky, D. Kahneman. 1974. Judgment under uncertainty: Heuristics and biases, *Science*, Vol. 185, pp. 1124-1131.

[17] A. Tversky, D. Kahneman. 1982. Judgment under uncertainty: Heuristics and biases, in *D. Kahneman, P. Slovic, A.Tversky (Eds.) Judgement under uncertainty: Heuristics and Biases*, Chapter 11, pp. 3-20, Cambridge University Press

[18] Haiquin Wang and Marek J. Druzdzel, 2000. User Interface Tools for Navigation in Conditional Probability Tables and Elicitation of Probabilities in Bayesian Networks, in *Proc. 16th Conference on Uncertainty in Artificial Intelligence*, pp. 617—625.

[19] Judea Pearl, 1988, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, San Mateo, California, ISBN 0934613737.