

# In Search of Best Practices for the Use of Natural Language in the Development of High-Consequence Systems

Kimberly S. Hanks, John C. Knight  
*Department of Computer Science*  
*University of Virginia*  
*151 Engineer's Way*  
*Charlottesville, VA 22904-4740*  
*{ksh4q|knight}@cs.virginia.edu*

## 1. Introduction

Natural language is ubiquitous in any software development process. Regardless of how much structure, whether from modelling languages, graphical depictions, outright formalism, or other mechanisms, is used in order to scaffold the organization of concepts during the process, natural language is the glue by which any of these representations is constructed, manipulated, and given meaning. All disciplines of engineering have best practices for the invocation and use of various strategies and tools, but none exists for the use of natural language in engineering of even the most critical software, despite the degree to which natural language pervades and influences this process. Recognition of this influence, particularly of the dangers of poor use, is accumulating, and we have set ourselves the task of exploring the issue of just what might go into the determination of best practices for the use of natural language in the development of software generally, and of high-consequence software specifically.

## 2. Why We Need Best Practices for the Use of Natural Language

Incidents and accidents that can be attributed to software failure often result in tragedies and other losses. The need to learn from these incidents and accidents and thereby inform the software process in the service of safety and dependability grows more critical as software systems become more complex and the ways they can fail become less intuitive. This need mandates the development of a more rigorous and systemic approach to forensic software engineering.

In exploring the issues to be addressed in order for the development of a systemic approach to forensic software engineering to be feasible, Johnson recognized that the proliferation of assumption and other communicative prob-

lems throughout the software process were recurring themes that were turned up in existing investigations of accidents. Importantly, he also demonstrated that these same deficiencies that plagued the process also plagued the reports and recommendations resulting from these investigations [6]. These deficiencies raise two issues. First, the potential for assumption and other miscommunication during the development process, especially in the early stages of a software project, can allow invalid conceptions of elements of the system to enter and persist, possibly leading to failures. Hayhurst and Holloway, among others, have argued that requirements is a communication problem, and thus that poor requirements result from poor communication [5], and Lutz has shown that the majority of safety-critical errors in the systems she examined were introduced at the requirements stage [7], rendering the problem that much more significant. Further, unstated or unclear motivations for requirements decisions impair the ability to analyze causes. Second, the potential for assumption and other miscommunication in the reports and recommendations resulting from investigations of incidents and accidents that derive from such failures can render such documents of little use. For example, if the analyses contained in a report are based on misconception, then a valid analysis has escaped recognition, and if the recommendations suggest ideals that are assumed to be achievable but are in reality impossible (as documented in [6]), then time and energy that could be applied to exploring new avenues for progress is likely to be wasted in the service of unattainable perfection.

To these we add a third issue: there exists the same potential for assumption and miscommunication in the statements of guidelines that prescribe the activities to be undertaken and the artifacts to be produced during incident and accident investigation and reporting. In contrast with the requirements for a software system or the reports resulting from investigations, guidelines represent meta-statements; they define the form and content of a class of

instances, whereas requirements and reports are instances of classes. The purpose of these meta-statements is in large part to standardize the results of investigations, such that, as a data set, the results can be compared with one another and analyzed for trends. In other words, the intended value of guidelines is that they predictably produce artifacts with properties that are useful to forensic software engineering. The potential for assumption and miscommunication in such guidelines impairs the likelihood that, for example, two different investigation teams will come to substantially the same conclusions while following the prescribed process, that is, this potential impairs predictability and the value of the resulting documents as a data set. This creates an additional area of focus within a systemic view of forensic software engineering, in which reduction of the potential for assumption and miscommunication in investigation and reporting guidelines is a necessary task if the big picture and the role of communication throughout it are to be improved.

### 3. What We're Doing About It

In previous work, we examined how the ways in which humans innately use natural language renders statements of requirements incomplete, inconsistent, and open to misinterpretation [3]. To accomplish this analysis, we exploited results from cognitive linguistics that detail the ways in which humans organize and communicate conceptual information. We extended this model to account for the breakdown that occurs in communication of information across boundaries of domain expertise, breakdown that is implicated as a major limiting factor of the quality of large and complex software systems [1].

Miscommunicated requirements, as noted above, are themselves a detriment to forensic software engineering [6]. The model by which we analyze and characterize communicative breakdown in requirements can also be applied to investigation and reporting guidelines, as well as to the reports and other documents that are generated, and we are currently conducting work to extend application of the model in this direction [4]. Implications of the model suggest ways to improve the use of natural language in these as well as other areas of the process, and experimental results bear out the value of an artifact we designed to systematically cope with semantic incompleteness, inconsistency, and ambiguity [2].

Attention to the focal areas of requirements, and accident investigation and reporting, which have significant influence respectively on the quality of software artifacts produced for high-consequence systems as well as the process for producing them, combines into a bigger picture: how to deal with natural language and its effects in the production of these systems. We have brought to bear results from linguistics in analyzing recurring problems of miscommunication and they suggest a battery of anchors around which we can shape our use of natural language. These anchors have already proven useful in early application, and as we extend and further evaluate them, we are slowly piecing together a picture of how to use natural language simultaneously to its smallest detriment and greatest value across all phases of development of systems that need to be safe and dependable. In other words, we are formulating a notion of best practices for the wielding of this necessary tool, whose misuse can result in disaster.

### 4. References

- [1] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31(11):1268-1287, 1988.
- [2] K. Hanks and J. Knight. An experiment in applying linguistic insight to improve requirements. Submitted to *10th International Symposium on the Foundations of Software Engineering*, Charleston, SC (November 2002).
- [3] K. Hanks, J. Knight, and E. Strunk. Erroneous requirements: a linguistic basis for their occurrence and an approach to their reduction. *Proceedings of the 26th Annual IEEE NASA Software Engineering Workshop*, pages 115-119, Greenbelt, MD (2001).
- [4] K. Hanks, J. Knight, and C.M. Holloway. The role of natural language in accident investigation and reporting guidelines. Submitted to *2002 Workshop on the Investigation and Reporting of Incidents and Accidents*, Glasgow, Scotland (July 2002).
- [5] K. Hayhurst and C.M. Holloway. Challenges in software aspects of aerospace systems. *Proceedings of the 26th Annual IEEE NASA Software Engineering Workshop*, pages 7-13, Greenbelt, MD (2001).
- [6] C. Johnson. Forensic software engineering. *Proceedings of 19th International Conference SAFECOMP 2000*, pages 420-430, Rotterdam, Netherlands (2000).
- [7] R. Lutz. Analyzing software requirements errors in safety-critical, embedded systems. *Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 126-133, San Diego, CA (January 1993).