References on Software Simplification

Edward S. Lowry 17 June 2013
Bedford, Mass.
eslowry@alum.mit.edu

The following referenced documents can help evaluate software language technology deficiencies. At present all safety-sensitive software is written in language whose simplicity of expression is poor compared with what was designed at IBM in the early 1970s and implemented at Digital Equipment Corp in the early 1980s. The resulting extraneous complexity has probably contributed to a variety of transportation disasters. Students everywhere are routinely taught how to arrange pieces of information by educators who are unaware of pieces of information that are well designed to be easily arranged.

REFERENCES

[1] E. S. Lowry, PROSE Specification, IBM Poughkeepsie Laboratory Technical Report TR 00.2902, Nov 23 1977.
This 69 page document describes the design an of an integrated programming and data base language which allows freely intermixed singular and plural expressions with rich data structures and a relatively natural language style. In retrospect, it could be viewed as a substantial improvement that introduced general plural expressions while increasing the richness of data structures. In contrast, languages like APL and SQL provided plural expressions at the cost of reduced data structure flexibility. Currently used languages do not succeed nearly as well in combining those capabilities. Success appears to depend on choosing elementary data structures from a narrow set of possibilities. Technical content of this document was distributed internally at IBM in 1974. It can be copied freely. A preliminary version was distributed in 1973.

[2] E. S. Lowry, DAWN Language Reference, Dec 17, 1982, 65 pages.
This describes the language (variously called Dawn, Keep, and Shannon) as refined and implemented at Digital Equipment Corp by 1983. It is copyrighted and there are only a few copies. The system was further developed as a multi-user data base system until 1989 but those documents are proprietary to DEC (now HP). About 20 people had some experience using the language.

Those 2 documents can be used to make a solid case that there has been a long delay in improving the simplicity of expression in software language. Using those documents to verify the delay is awkward, but practical. Coding examples can help make the severity more evident. These documents support the claim of a 40 years delay on the above 6 leading edges of design.

[3] E. S. Lowry, "Inexcusable Complexity for 40 years", June 2013, 8 pages.
users.rcn.com/eslowry/inexcus3.pdf .
Abstract: Eliminating contamination and understanding basic structures are high priorities in many technologies. Both have been neglected in software technology. Language for expressing precise information has had needlessly deficient design for over 40 years, probably on all six of the following leading edges:

1. Flexibility of data structures.
2. Simplicity of expression.
3. Generality of subject matter.
4. Modularity of elementary information building blocks.
5. Fluency of precise expression.
6. Durability of design of core language semantics.

A false dichotomy has prevented combining wide use of simple plural expressions with flexible data structures. Incorporating iteration connections into basic data objects circumvents the dichotomy.

[4] E. S. Lowry, "Toward Perfect Information Microstructures", users.rcn.com/eslowry/tpim.pdf , Jan. 2010, 26 pages.
This is a plausibility theory that thoroughly eliminating extraneous complexity leads to convergence on a unique permanent optimum structure for atoms of information that are designed to be easily arranged (or highly modular). The first 4 hypotheses which argue that convergence does happen are fairly straightforward. The others make a case for what the final optimum will be. This theory appears to have large disruptive potential and has received no significant technical evaluation.

[5] E. S. Lowry, "Software Simplicity and Safety, Thwarted for 40 Years", 31st Digital Avionic Systems Confernce (DASC) Proceedings, 18 Oct. 2012, Williamsburg VA, Powerpoint slides only.

[6] "Position Statement: Upgrading the National Airspace System (NAS)"
Adopted by the IEEE-USA Board of Directors, 17 February 2012
www.ieeeusa.org/policy/positions/NatlAirspaceSystem0212.pdf
Its fifth recommendation includes:
"Currently, all safety-related software has been developed using
computer language technology that has poor simplicity of
expression, compared to what was designed in the early 1970s, and
implemented in the early 1980s, by leading computer vendors [...]."

[7] E. S. Lowry, "Optimum data objects for technical literacy", Educational Technology & Society, Vol 2 No 1, Jan 1999. www.ifets.info/journals/2_1/ed_lowry_short_article.html .
This published article sketches some ideas from [4].

[8] E. S. Lowry, "Technical Fluency Stifled for Decades", January 2010, 15 pages, users.rcn.com/eslowry/stifled.pdf .
This discusses how the long delay developed. Sound policy for fixing the problems requires understanding the ways in which technical progress has been delayed over the past 40 years. References are included to unreasonable perspectives which may have delayed progress.

[9] E. S. Lowry, "Software Simplicity, and hence Safety -- Thwarted for Decades", 2004 International Symposium on Technology and Society (ISTAS'04).
A published early version of [8].

[10] E. S. Lowry, "Formal Language as a Medium for Technical Education", Proceedings of ED-MEDIA 96, p407, AACE, June 1996. See also users.rcn.com/eslowry/edmedium.htm This discusses potential application of improved precise language in education.

[11] E.S. Lowry, "Data Processing System Having a Data Structure with a Single Simple Primitive", US Patent No. 5,664,177, Sept. 1997.
[12] "In Re Edward S. Lowry", 32F.3d 1579 (Fed. Cir. 1994)
This is a patent on some of the data structure ideas and a precedent setting appeals court case that awarded the patent. The effect has been to delay progress. The patent should expire in 2014.

[13] Correspondence with U.S. Senator John Kerry and several U.S. government agencies from Dec 2000 to Aug 2005. Letters inquiring about available software language capabilities were forwarded to several agencies. The responses and non-responses were evasive.

COMPARATIVE CODING EXAMPLES

[14] A series of 10 short expressions were translated from Sequel2 (later SQL) into Shannon. They took 130 tokens in Sequel2 and 99 tokens in Shannon. The Shannon version is in the appendix to [4] along with a reference to the original Sequel2. The first 7 of those were also translated from 68 tokens of Shannon into 433 tokens of C++ which appear in appendices to [3] and [8].

[15] An Accounts Payable program sold by DEC was originally written in 140 pages of Dibol (a poor language) to run on a PDP-8 computer (a 12KB machine). It was translated into 27 pages of Dawn (Shannon) to run on a DEC VAX. The Dibol is not available. The Dawn version is available in hard copy.

[16] ML is a language suggested by 2 computer science professors as providing good simplicity of expression. The book "ML Primer" by Ryan Stansifer, Prentice Hall 1992, includes an example describing the Klondike Solitaire card game on pages 110-115.
I translated the textbook example from 1416 tokens of ML into 475 tokens of Shannon (using a few unimplemented features that make little difference).

[17] A single example is given by:

6 = count every state where populatn of some city of it > 1000000

This example illustrates an expression with little remaining extraneous complexity. Software developers will probably see quickly that translating it into any language currently used for safety-sensitive software will force objectively inexcusable complexity to be generated. The expression is written in the general programming, data base, and modeling language specified in reference [2]. Facilities capable of executing [2] including the above expression were operational in 1982. This example taken in context tells a short sharp story about long term neglect of simplicity and safety in software.

SOME EARLY EXPLORATIONS OF THE ABOVE IDEAS

[18] E. S. Lowry, "Proposed Language Extensions to Aid Coding and Analysis of Large Programs", IBM Poughkeepsie Lab Technical Report TR 00.1940-1, Nov 21, 1969, 22 pages. This also appears in NATO Science Committee Conference, "Techniques in Software Engineering", Rome, Italy, 27 to 31 Oct 1969, Working Material Vol II, Sept 1969.

[19] E.S. Lowry, "Accurate description of system structure - A new standard of language quality", Proceedings of the Digital Equipment Computer Users Society, Vol 5, No2, 1978, pg833.

[20] E.S. Lowry, "Nodes and Arcs, The Ideal Primitive Data Structures", DEC/TR-114 November 1980, Digital Equipment Corporation, 26 pages. Abstract only in ACM'80 Proceedings.

[21] E.S. Lowry, "Toward an Optimum Language Data Model",
Computer Standards & Interfaces 13(1991) p105-108.