Richard Gisselquist

Engineering in Software

Software development is similar to building bridges. Technical expertise is applied to build something that may be confidently used where expertise is lacking. A person crosses a bridge without measuring the beams and repeating the stress analysis arithmetic. A person uses a compiler without verifying the symbol table algorithm.

It is this application of expertise that causes the fields of both bridge building and software development to be areas where engineering principles are applicable. It is the cost of failure that makes these principles required.

However, software is different from concrete and steel. The obvious differences are shape and strength. Concerning shape, software is like a cartoon. As an animated drawing can assume unrealistic shapes, software supporting animation can produce these shapes. Concerning strength, software does not suffer from fatigue or vibration. If there is sufficient time and memory, an algorithm in software can function correctly far beyond where it can function well. For example, imagine the bridge-loading analogue of a linear search through the O.E.D. with a Turing machine.

Therefore, producing software is an engineering endeavor at the level of responsibility and ethics. But it is not an endeavor that can make use of the traditional engineering tools—slide rules and tables of yield strength.

OO and CASE Are Not Engineering

There are bridges made of steel and bridges made of concrete. There are also bridges made of wood. If the correct stress calculations can be performed, a highway bridge can be built of paper mache.

To apply the analogy to software, good software can be written in C or Pascal, assembler or Fortran. The choice of language, like the choice between steel or wood, does not mean good engineering.

Similarly, there are bridge construction projects that use a crane mounted on a pair of barges, or that use scaffolding extending out from the center pier. For one suspension bridge, the first line across the river was carried by a rowboat; for another, it was carried by a kite.

As applied to software, some projects are designed on the blackboard, others using computer-based design tools. Some projects purchase outside tools, others write in-house source librarians, testbeds, and validation facilities.

In each case, the methodology may represent sound engineering practice. But the methodology—by itself—does not assure good engineering for the finished product.

Management Is Engineering

If engineering in software is not calculators and strength-of-materials tables, is not advanced languages and tools, then what remains? For starters, management remains. There are two responsibilities that fall on management.

The first is feasibility. Management is responsible for accepting the challenges while refusing the impossibilities. Those doomed projects that will be late, over-budget, and unreliable must be replaced by alternatives that will generate customer satisfaction.

The second management responsibility is the nurturing of the organism that is the software development capability: the people and the process. Good employees should be hired and kept. Helpful tools should be provided and supported. The design documents and code

should be subject to appropriate reviews.

The resulting software may have an author's name on the design document, and may have several coders' names on the subroutines, but the manager's name will be on the whole.

Design and Implementation Are Engineering

In the bridge example, the design specifies everything that will be in place when the bridge is used. The implementation of the bridge, called construction, is the activity required to complete the bridge. The design and implementation are activities that differ at a conceptual level.

With software, design and implementation are the same activity—translation. A software project moves from desire to silicon by a process of repeated translations: from specifications to design to detail design to source code.

The danger of losing control of the logic exists at each translation step. The prize of possibly adding value to the software through and excellent translation exists at each step.

Good engineering in software consists of accurate and appropriate translations.

Engineering Is Use

The owner of a steel bridge should paint the bridge to prevent rust, inspect the bridge on a regular basis, and post accurate load limits.

Software will never need to be painted, but it can fail through misuse. Software owners should employ competent operators, monitor the performance on a regular basis, back up the critical data, and provide adequate memory space and processing power.

Hackers celebrate when they can use the campus computer to ring the bells in the college chapel at two in the morning.

Scientists celebrate when they have successfully sent a monkey into space.

Engineers do not celebrate until they can walk across the completed bridge, holding their children's hands. C

Richard Gisselquist (rig@cray.com) is a senior programmer analyst at Silicon Graphics in Eagan, Minn.