Lehman’s Laws and related background

Perdita Stevens

School of Informatics
University of Edinburgh
Various experts have asserted that most of the cost of software ownership arise after delivery, i.e. during “maintenance”.


But software doesn’t wear out?!?!

No, but it gets

▶ fixed (corrective maintenance),
▶ adapted to changing needs (adaptive maintenance),
▶ improved in performance or maintainability (perfective maintenance)
▶ improved by fixing bugs before they activate (preventive maintenance)

[ISO/IEC 14764, following Swanson]
What should we think of this?

Success: tells of flexible systems that needn’t be thrown away?
Failure: tells of systems that aren’t correct or flexible as built?
Whatever... figures like these do tell us that how maintenance is done is important: doing it better may save money.
(And doing it less may too, of course.)
Lehman’s laws

Manny Lehman, the “Father of Software Evolution”, wrote many papers from the mid 70s onwards, proposing “Laws of Software Evolution” for “E-type systems”.

Systems classified into:

- S-type: formally specified and verified; static by definition
- E-type: real-world system
### Lehman’s laws (adapted from 2001 talk by MML)

<table>
<thead>
<tr>
<th>I</th>
<th>Continuing Change</th>
<th>An E-type system must be continually adapted else it becomes progressively less satisfactory in use</th>
</tr>
</thead>
<tbody>
<tr>
<td>II</td>
<td>Increasing Complexity</td>
<td>As an E-type system is evolved its complexity increases unless work is done to maintain or reduce it</td>
</tr>
<tr>
<td>III</td>
<td>Self regulation</td>
<td>Global E-type system evolution processes are self-regulating</td>
</tr>
<tr>
<td>IV</td>
<td>Conservation of Organisational Stability</td>
<td>Average activity rate in an E-type process tends to remain constant over system lifetime or segments of that lifetime</td>
</tr>
<tr>
<td>V</td>
<td>Conservation of Familiarity</td>
<td>In general, the average incremental growth (growth rate trend) of E-type systems tends to decline</td>
</tr>
<tr>
<td>VI</td>
<td>Continuing Growth</td>
<td>The functional capability of E-type systems must be continually enhanced to maintain user satisfaction over system lifetime</td>
</tr>
<tr>
<td>VII</td>
<td>Declining Quality</td>
<td>Unless rigorously adapted to take into account changes in the operational environment, the quality of an E-type system will appear to be declining as it is evolved</td>
</tr>
<tr>
<td>VIII</td>
<td>Feedback System</td>
<td>E-type evolution processes are multi-level, multi-loop, multi-agent feedback systems</td>
</tr>
</tbody>
</table>
Criticism of Lehman’s laws

“Laws”?

Based on data?

Contentful?
Terminology

Legacy system
Reverse engineering
Reengineering
Program comprehension
Evolution
Maintenance: corrective, adaptive, perfective (Swanson)
Legacy systems

A system which still has value, but which significantly resists modification and evolution.

Stereotypically *old* – but that can mean 5 years.

Problems include:

- architectural degradation
- reliance on unmaintained software or hardware
- loss of expertise
- not designed for evolution.
So what to do?

Basically three options:

- Soldier on
- Reengineer
- Scrap

The attempt to understand the system is an essential part of the decision process.
A few sources

The Lehman talk I used, *Software Evolution: from Observations to Theory* and the position paper *Laws of software evolution revisited* are both available from http://www.doc.ic.ac.uk/~mml/feast2/papers.html