CS 428 – Webster readings #6

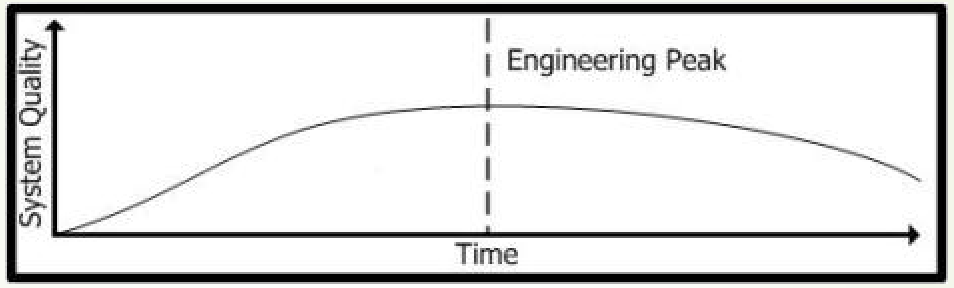
Winter 2019

Bruce F. Webster

Getting Technology Lifecycles in Sync (Baseline, 2009) [Link]

- Certain cycle mismatches for organizations looking to adopt new technologies – keep your eyes open for these:
- **Firefly**: initial release is inadequate; you wait for the next; the next release is not quite functional enough, either; rinse and repeat until you finally give up.
- **Underdone**: new release of established technology that's not quite ready for production use
- **Conveyor Belt**: limited lifespan – challenge to make use of it and then migrate off before it goes away.
- **Landfill**: great promise (hype), but dead on (or soon after) arrival.

The Arc of Engineering (Baseline, 2008) [Link]

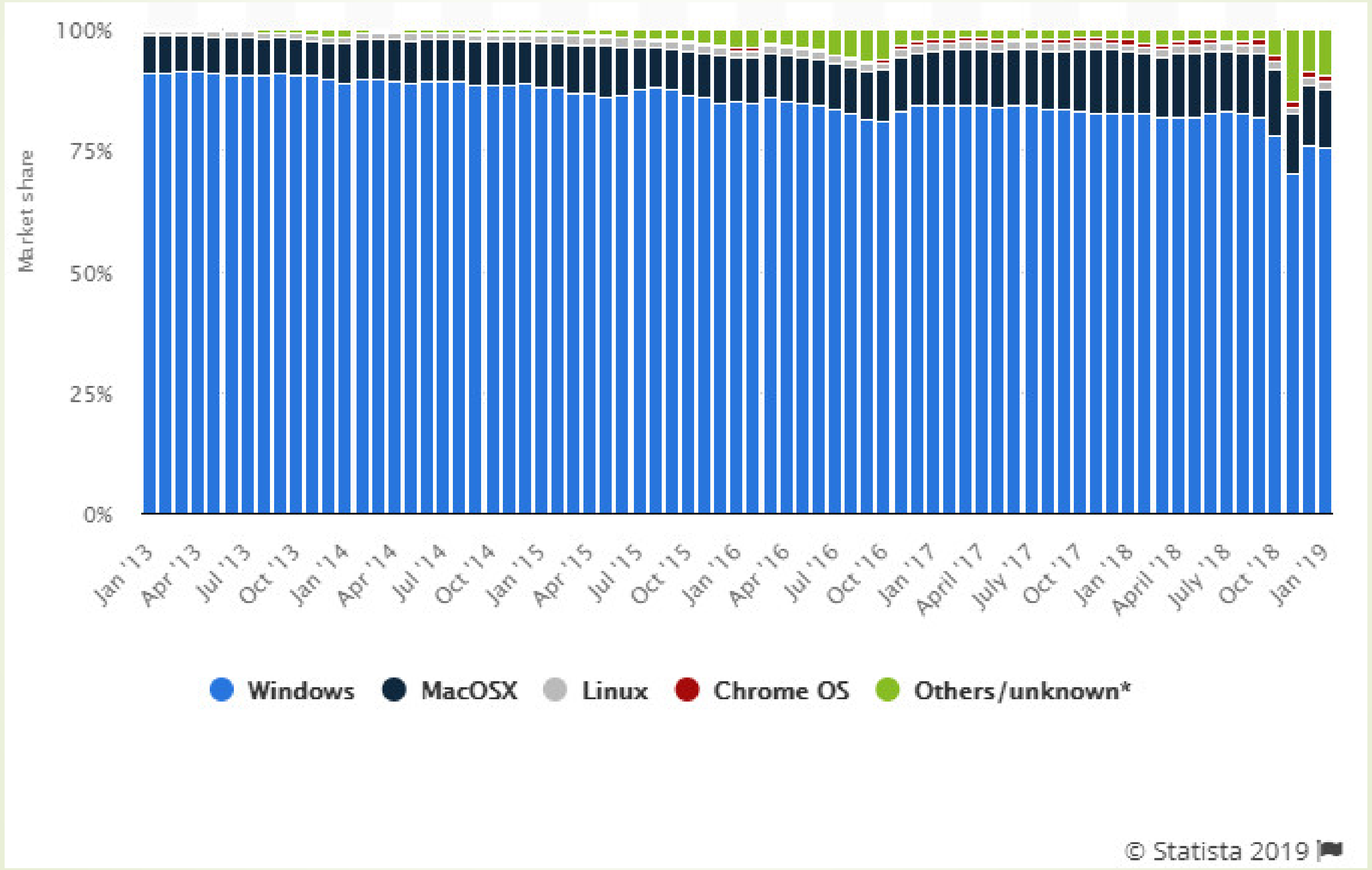


- System/product quality rises over time to a peak value and then starts to decline. Why?
  - The developer loses conceptual control of the system (bloat, loss of personnel, organizational changes)
  - Software rot sets in (piecemeal changes, growing incompatibilities w/external interfaces)
  - The enhanced system finally outgrows its original foundation
  - Market or business needs shift beyond the product's fundamental design
  - The developer beings to add "blue sky"/"kitchen sink" enhancements
  - Backward compatibility is maintained at all costs

Microsoft Windows Forever and Ever? (Windows Magazine, 1996) [Link]

- Me, in 1996: "As impossible as it might seem, Windows may still be dominant in 2025, nearly 30 years from now. It may look and work a bit differently, just as phones, TVs, and cars from 30 years ago do, but the principles will be the same. Our grandchildren will wonder about the quaint relics of terminology and work flow (when was the last time you actually put gloves in a glove compartment?), but they'll be able to clearly see the inheritance from MS-DOS/Windows to whatever they use."
- As of Jan 2019 – see next slide.

Windows Forever and Ever? (cont.)



Microsoft Windows Forever and Ever? (cont.)

- It seems that the first sufficiently adequate technology in a given sphere usually gains broad acceptance and entrenches itself. There may be some contention for standards early on, but that eventually settles out. And once the solution gains acceptance, the rest of civilization reinforces it–through market forces, legal standards, competitive pressures, and economic incentives.
- Once the technology is entrenched, the focus is then on refinement and slow upgrading of the existing technology, not on radical innovations and wholesale replacements. This has been shown time and again in major areas of applied technology: communication (phones), transportation (cars), and so on. And the same thing is happening now with information technology.
- The sheer scale of adoption of Microsoft technology will provide tremendous momentum to keep thing moving in the same direction. The investment in hardware, software, market standards, training, business process, development expertise, custom applications, and deployed environments all argue against any broad changes, even those introduced by Microsoft. That investment grows year by year and will dominate more, not less, as time goes on.

Bonus article: Septic Code [2013]

- **Definition**: some portion of the source code created to date is so bad and has such a negative impact on other code that relies upon it that the project will never stabilize until that source code is cut out of the project and thrown away, and brand new source code meeting professional standards is created in its place.
- **Example**: *A lot of it is bad old code that BigFirm didn't have time to rewrite two years ago, but now is five times its original size and even worse. One consultant said he took a code listing, picked pages at random, and found problems on every page he selected. There is pervasive hard coding of what should be adjustable parameters or at least meaningfully named constants (e.g., # of [key items] hard-coded throughout with the literal value '3', a constant named 'ninety_eight'). Builds take all night. App releases don't run acceptably, if at all, in a production environment. Developers check in files that won't even compile.*