Feature creep

Feature creep, **creeping featurism** or **featuritis** is the excessive ongoing expansion or addition of new features in a product,[1] especially in computer software and consumer and business electronics. These extra features go beyond the basic function of the product and can result in software bloat and over-complication, rather than simple design.

Contents

Causes

The most common cause of feature creep is the desire to provide the consumer with a more useful or desirable product, in order to increase sales or distribution. However, once the product reaches the point at which it does everything that it is designed to do, the manufacturer is left with the choice of adding unneeded functions, sometimes at the cost of efficiency, or sticking with the old version, at the cost of a perceived lack of improvement.

Another major cause of feature creep might be a compromise from a committee which decides to implement multiple, different viewpoints or use cases in the same product. Then, as more features are added to support each approach, it might be necessary to have cross-conversion features between the multiple paradigms, further complicating the total features.

Characteristics

Feature creep is one of the most common sources of cost and schedule overruns.[2] It thus endangers and can even kill products and projects.

Control

There are several methods to control feature creep, including: strict limits for allowable features, multiple variations, and pruning excess features.

Temptation of later feature creep may be avoided to some degree by basing initial design on strong software fundamentals, such as logical separation of functionality and data access. It can be actively controlled with rigorous change management and by delaying changes to later delivery phases of a project.[3]

Another method of controlling feature creep is to maintain multiple variations of products, where features are kept limited in some variations. Because the ever-growing, ever-expanding addition of new features might exceed available resources, a minimal core "basic" version of a product can be maintained separately, to ensure operation in smaller operating environments. Using the "80/20 Rule" the more basic product variations might support the needs of about "80%" of the users, so they would not be subjected to the complexity (or extra expense) of features requested by the other 20% of users. The extra features are still available, but they have not crept into all versions of the products.

At some point, the cost of maintaining a particular subset of features might become prohibitive, and pruning can be used. A new product version could simply omit the extra features, or perhaps a transition period would be used, where old features were deprecated before eventual removal from the system. If there are multiple variations of products, then some of them might be phased out of use.

Consequences

Expansion of scope

Occasionally, uncontrolled feature creep can lead to products far beyond the scope of what was originally intended; this is known as scope creep. However, a more common consequence of feature creep is a delay or cancellation of the product, which may become more expensive than was originally intended.

Delays

Often, a reasonably feature-complete software project, or one with moderate amounts of feature creep, can survive and even thrive through many iterations, but its successor release may suffer substantial delays once a decision is taken to rewrite the whole code base in addition to introducing new technologies. For example, Microsoft's Windows Vista was planned to be a minor release between Windows XP and its successor codenamed Windows "Blackcomb", but after adapting more and more features from Blackcomb (many of which were eventually cancelled), Vista turned out to become a major release which took five years of development.

A similar fate was suffered by Netscape 6, which was originally supposed to be Netscape 5. The 1998 decision by Netscape Communications to open-source its Netscape Navigator browser and Communicator Internet suite (both code-named Mozilla) soon made it obvious that the underlying code was too difficult, and required a complete rewrite of Mozilla, which fostered the creation of the Mozilla application framework. This caused significant delays, Netscape 5 was skipped, and the company was purchased by AOL. The subsequent release of Netscape 6.00 in 2000 was widely criticized as alpha-level code, and the project reached stability by Netscape 6.1 in 2001, three years after the decision to rework the Internet suite. By that time, Microsoft's Internet Explorer browser had long-eclipsed Netscape in usage share, which had diminished to single digits.

Even after reaching stability and attaining some necessary new features, the open-source Mozilla Application Suite (then named just Mozilla), on which AOL built Netscape, was viewed as "bloated". Just a year later, a group of Mozilla developers decided to separate the browser component, which eventually became Firefox.

Double Fine Adventures' Kickstarter project *Broken Age* is another example of a project being delayed by feature creep. Originally supposed to have a release date of October 2012, the first half of the game was released in January 2014 while the second half followed late April 2015, and required two separate funding rounds to complete.[4]

Feeping creaturism

Feature creep combined with short deadlines will often lead to a "hacky solution". The desired change may be large enough to warrant a redesign of the existing project foundation, but deadline pressure instead requires developers to just "make it work" with a less elegant approach. The humorous spoonerism "feeping creaturism" was coined to emphasize a developer's dislike of this situation,[5] personifying the scope-crept product as "a misshapen creature of hacks ... prowling about in the dark",[6] and the harbinger of more creep to come.[7] ("Feeping" is a jargon synonym of "beeping".)[8]

See also

- Creeping elegance
- Design document
- Greenspun's tenth rule
- KISS principle
- Minimalism
- Mission creep

- Overengineering
- Scope creep
- Second-system effect
- Software bloat
- Plug-in (computing)
- Unix philosophy
- Zawinski's law of software envelopment

References

1. J.M. Sullivan (8–10 June 2005), "Impediments to and incentives for automation in the Air Force" (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1452719), *2005 International Symposium on Technology and Society*, pp. 101–110, doi:10.1109/ISTAS.2005.1452719 (https://doi.org/10.1109%2FISTAS.2005.1452719)
2. Davis, F.D.; Venkatesh, V. (February 2004), "Toward preprototype user acceptance testing of new information systems: implications for software project management" (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1266852), *IEEE Transactions on Engineering Management*, 51, IEEE Transactions on Engineering Management, issue 1 (1): 31, doi:10.1109/TEM.2003.822468 (https://doi.org/10.1109%2FTEM.2003.822468), ISSN 0018-9391 (https://www.worldcat.org/issn/0018-9391)
3. Kenneth S. Norton (2001), *Applying Cross-Functional Evolutionary Methodologies to Web Development* (https://books.google.com/books?id=Ak5slktYul8C), paper in Web Engineering: Managing Diversity and Complexity of Web published by Springer, ISBN 3-540-42130-0
4. Double Fine splits Broken Age in half to fund completion (http://www.gamasutra.com/view/news/195509/Double_Fine_splits_Broken_Age_in_half_to_fund_completion.php), By Kris Ligman, 2013-07-02, Gamasutra
5. feeping creaturism (http://foldoc.org/feeping+creaturism), 2016-05-27, FOLDOC.org - The Free On-line Dictionary of Computing
6. Raymond, Eric S.; et al. (December 29, 2003). "feeping creaturism" (http://catb.org/jargon/html/F/feeping-creaturism.html). *The Jargon File*. Ver. 4.4.7. Retrieved June 20, 2017.
7. Raymond, Eric S.; et al. (December 29, 2003). "feeping creature" (http://catb.org/jargon/html/F/feeping-creature.html). *The Jargon File*. Ver. 4.4.7. Retrieved June 20, 2017.
8. Raymond, Eric S.; et al. (December 29, 2003). "feep" (http://catb.org/jargon/html/F/feep.html). *The Jargon File*. Ver. 4.4.7. Retrieved June 20, 2017.

External links

- Creeping Featuritis (https://web.archive.org/web/19961130075228/http://c2.com/cgi/wiki?CreepingFeaturitis), ContentCreationWiki (registered on October 23, 1995 (https://web.archive.org/web/19961130075228/http://c2.com/cgi/wiki?CreepingFeaturitis) at the latest.)
