

Building Software, Like Building Airlines, is a Feat of Engineering.  
Should We Build Software Like We Build Airlines?

Conley Read  
[cread@cs.ucr.edu](mailto:cread@cs.ucr.edu)

March 1, 2005

CS 245, Software Engineering  
Dr. Doug Tolbert

# Table of Contents

Table of Contents .....	2
Introduction.....	3
Which is the bigger beast? .....	3
The Boeing 747 airliner .....	3
The Windows™ XP operating system .....	3
The Linux Kernel .....	4
Engineering the 747 .....	4
How to make an airliner? .....	4
Why are airliners reliable? .....	5
Software Engineering.....	5
What is software?.....	5
Why is software unreliable?.....	5
Process similarities... and differences .....	6
The future of Software Engineering.....	6
Software engineered like an airliner .....	6
Would software be more reliable? .....	6
Software without innovation? .....	6
Who said, “No Silver Bullet”? .....	7
References.....	8

## Introduction

In the field of software engineering, one often encounters the contention that software would be more reliable if it were built in a process like that used to build airliners. In this short paper, we evaluate the truth of the notion in a variety of useful contexts. We set up a few timelines to normalize development events in the history of Airliners and examine our comparison of similarities with respect to recent statistics.

The design and construction of airliners like the Boeing 747, which we will use as our primary example, is characterized by strict process implementation and regulation in every step. Would software be more reliable if it were subject to similarly designed and purposed processes?

I propose that the field of software engineering *can* be made more like aeronautical engineering. We will give an overview of exactly how this may be achieved. The question remains. Is this a good direction for software engineering to move in? I do not think it is. We identify what who does right, and when, the implications on innovation, and what we can learn from our beloved Boeing.

### *Which is the bigger beast?*

#### **The Boeing 747 airliner**

The Boeing 747 wide body airliner entered regular air service in 1970 seating up to 490 passengers for transatlantic flight. This aircraft design heralded the “Jumbo Jet” era. The 747-model airliner, with its updated avionics and interior is definitely a few versions down the road from where it started in January 1970, but it is still in use by Airlines around the world.

The Boeing 747 is assembled from greater than six million parts supplied by over 1000 vendors located all around the globe. Three million of these parts are fasteners including pins, bolts, and over one and a half million rivets alone.

These fasteners support the almost one million pound designed take-off weight of the Boeing 747. During repair and maintenance cycles, each of these three million fasteners is inspected individually for wear and defect to ensure the safety of the millions of passengers that depend on its reliability for their transportation every year.

The Boeing manufacturing facility located in Everett Washington has a cult attraction of its own. Every year, approximately 120,000 visitors tour the plant and observe Boeing model 747s, 767s, and 777s in all stages of assembly in the Guinness Book of World Records “Largest Building in the World by Volume.”

Boeing Corporation employed 157,000 people in 27 states and around the world (2003).

#### **The Windows<sup>TM</sup> XP operating system**

The Windows<sup>TM</sup> XP operating system is often referenced for comparison of software development engineering to the accepted complexity of the aeronautical engineering field. Although the secrecy of

Microsoft and little public information about the engineering processes of its leading products are documented. There are many useful statistics of the latest incarnation of Microsoft's operating system, Windows™ XP.

There are many ways to measure the complexity of software. One of the most popular and time-tested is the measure of Source Lines of Code (SLOC).

Windows™ XP, after shipping in October 2001 consisted of about 40 million source lines of code. To put this number in perspective, it is helpful to note the relative size of the previous Microsoft Windows™ versions. Windows™ 95 had about 11 million lines of code, Microsoft Windows™ NT version 5.0 had about 20 million SLOC, and Microsoft Windows™ 2000 shipped with 35 million lines of code. With Windows™ XP service packs 1 and 2 adding a number of functionality upgrades, the installed Windows™ platform may stand at 41 million lines with the next announced version of Windows™ nearing the mark of 50 million total lines.

Microsoft Corporation employed 57,000 workers around the world in 2003, including over 10,000 employees in its Redmond, Washington headquarters.

## **The Linux Kernel**

To avoid leaving a widely heralded and increasingly important operating system out of the fray, we will include basic statistics for the Linux Kernel.

Recent versions of the Linux 2.6 Kernel have 5.7 million lines of code or more. It should be noted that comparing the Linux Kernel and Windows™ source lines of code numbers is like comparing apples to oranges. The Linux Kernel contains almost entirely core-operating functions only. Conversely, the Windows™ source base includes applications like Internet Explorer, Windows™ Media Player, and more. Due to this factor, it is very difficult to gauge correctly the size of the corresponding Windows™ Kernel source lines of code.

RedHat Corporation, producer of the commercial Linux distribution, RedHat, employed 641 people in 2004, a twenty percent increase in employee population. RedHat is growing fast too. In 2002-2003, employee population and expenditures increased by almost 60 percent.

## **Engineering the 747**

### ***How to make an airliner?***

In the late 1960s, Boeing was about the same size as Microsoft is today. With about fifty thousand employees and 1500 suppliers, they made the first 747. The early 747 had ¾ of the parts used today, only 4.5 million! Remember, they did this all with slide rules! It took longer and required more people on the engineering teams too.

More important than the creation of the Boeing 747 was the revelation that a process to ensure that every 747 was built the same as the last, perfectly. It is this process, perhaps a lesson learned during mass

production of the super fortress during World War II that has been admired by engineers around the world for decades.

### ***Why are airliners reliable?***

If you were following the numbers as you read, you will have noticed something striking about the number of parts used to construct the Boeing 747. Where software has routinely increased in complexity by up to 20 percent in one year, the Boeing 747 assembly complexity has only increased by 33 percent in 35 years. This is less than a one percent increase in complexity each year.

What is the cause of this huge complexity gap? How do aeronautics engineers maintain the 747 design and its reliability? Are the Boeing engineers meeting consumer demands?

## **Software Engineering**

### ***What is software?***

As it turns out, some people already know. The writers of the American Heritage Dictionary know. They agree that software is “the programs, routines, and symbolic languages that control the functioning of the hardware and direct its operation.” Easy enough.

We are already seeing a few reasons why software and airliners are built a bit differently. Earlier, you might have been saying, “airliners have software onboard too! Aren’t you just begging the question? Obviously Boeing got its flight control software right!”

Let us define two areas of software design so that we can continue. First, it’s “easy” to design software when the customer knows what they want. Flight control software is physics. And, right now? They are still dealing with hardware fault recovery. Boeing is, in this scenario, the customer. In a sense, there exists a “right,” correct way to implement flight control software. Second, there is design of software for customers who do not know what they want, or even what they require. This is the land where Microsoft Corporation is currently king. There is no “right” way, yet. The consumer will take all that they can get. They want it now. They want tomorrow’s technology, yesterday.

### ***Why is software unreliable?***

Software is primarily unreliable due to a small list of causes. Software can have programming errors. Software can be incorrectly specified. Software requirements can be incorrect. Software design changes can occur too often.

To get reliable software, we just take the negative sense of everything I have mentioned! Sounds easy, right? Of course, there are additional factors that can affect project outcomes, including people management, budget, and design feasibility. However, each of these is a common business problems encountered in each space of enterprise. Therefore, we will leave them unexplored here.

There is good news too. Some organizations do make highly reliable software. For one example, we can look at the NASA Onboard-shuttle group. They design the flight-control software for the NASA shuttles.

Not surprisingly, they develop their software in much the same fashion that Boeing manufactures its airliners! However, there are many trade-offs. Are the benefits worth it?

## **Process similarities... and differences**

We have started to see how the engineering processes differ between consumer software design and accepted airliner engineering and manufacturing processes. To a somewhat casual observer, the processes acceptable in each area may look very similar or possibly even identical, that observation is only on the surface. Perhaps this is because the leaders of software design looked to airliner manufacturing processes for a baseline highly effective design process. The design processes accepted in each are characterized by recognizably similar phases. The core phases include: specification, requirement, design change, and verification.

Additionally, these processes must incorporate ordinary people management and cost and complexity estimation.

Therefore, although the competing processes are not identical, contemporary software engineering definitely shares similarities with current airliner process engineering. What is the big deal? Why can't we, as software engineers, adopt aeronautical engineering techniques and gain the reliability advantage?

Unfortunately, there is an entire class of differences that would make this transition difficult.

## **The future of Software Engineering**

### ***Software engineered like an airliner***

In software engineering, we can learn a lot from other fields. Often, these "other fields" are our customers. It pays us to know our customers so that we can serve them better. Could we adopt aeronautical design practices? Definitely. Do we want to adopt them wholesale? Count me out.

As I mentioned before, there are a whole list of attributes that make our field unique. We want to drive towards reliability. Like our products, our evolution is important. To throw away all our effort to this point and accept a ready-made straightjacket would be like a child wanting to stay small... just to avoid growing pains.

### ***Would software be more reliable?***

Our field can lead the way in adaptation and in innovation. It is certainly possible for Microsoft to freeze modifications to its operating system and refuse unspecified design changes. Would Windows™ be more reliable? Yes! It would be stable, but also in a sort of stasis. Stasis is not good for business. Stasis is not good for innovation.

### ***Software without innovation?***

Honestly, our field knows how to build reliable software. Solid requirements and minimal design changes combined with continuous testing derive reliable software. Industry and consumers demand innovation. Where is the innovation in aerospace? It's not in the Boeing 747! The design for the 747 has remained relatively unchanged for 35 years. Our field has hardly been around that long!

### ***Who said, "No Silver Bullet"?***

Our study of the other engineering disciplines can only strengthen our field. We must adopt techniques that make software reliable and do not significantly affect innovation in the general case. We must understand the benefits of standard process so that we can design rock-solid software for our core products. As I initially proposed, the field of software engineering *can* be made more like aeronautical engineering. Initial review shows this direction to be an undesirable one.

With recent versions of Windows<sup>TM</sup> and Linux driving toward better reliability, we begin to see that reliability and innovation are not exclusive goals. In fact, with recent studies on the Linux Kernel we see innovation leading reliability on the horizon. The solution is to build software, not airplanes, just more reliably.

Frederick Brooks, author of the Mythical Man-Month, said, there's "no Silver Bullet." The situation is also the same here. Innovation has been the life-blood of our field and the industry that we support. For this very reason, I strongly oppose a wholesale adoption of airliner manufacture process and techniques. Software engineering is a young discipline facing many of the same problems faced before by others, but we face many new problems that make our decisions unique. We have to make good trade-offs. Sometimes we must learn the hardest way. When we do something first, there is no other way to learn.

## References

- [1] “Wonder how airplanes are being built?” Boeing Feature Release.  
[http://www.boeing.com/commercial/news/feature/evt\\_tour.html](http://www.boeing.com/commercial/news/feature/evt_tour.html)
- [2] “Peter Coffee on Apollo 11...” The Old Joel on Software Forum.  
<http://discuss.fogcreek.com/joelonsoftware/?ixPost=13020>
- [3] “Why does a new wing cost \$2 billion?” Airliners.net: Civil Aviation.  
[http://www.airliners.net/discussions/general\\_aviation/read.main/1952137/](http://www.airliners.net/discussions/general_aviation/read.main/1952137/)
- [4] “Apollo 11, the 747, and .Net? No way.” eWeek.  
<http://www.eweek.com/article2/0,3959,437228,00.asp>
- [5] “The Mythical Man-Month.” Frederick P. Brooks, Jr.
- [6] “Software Engineering.” Ian Sommerville. 7<sup>th</sup> Ed.
- [7] “They Write the Right Stuff.” Fast Company.  
<http://www.fastcompany.com/online/06/writestuff.html>