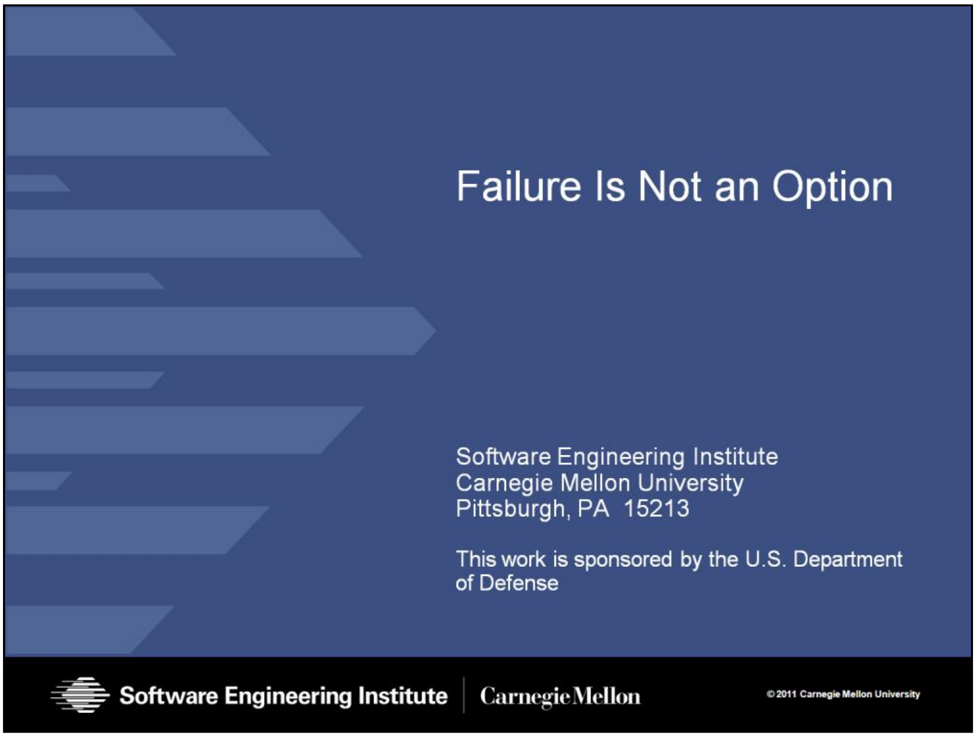
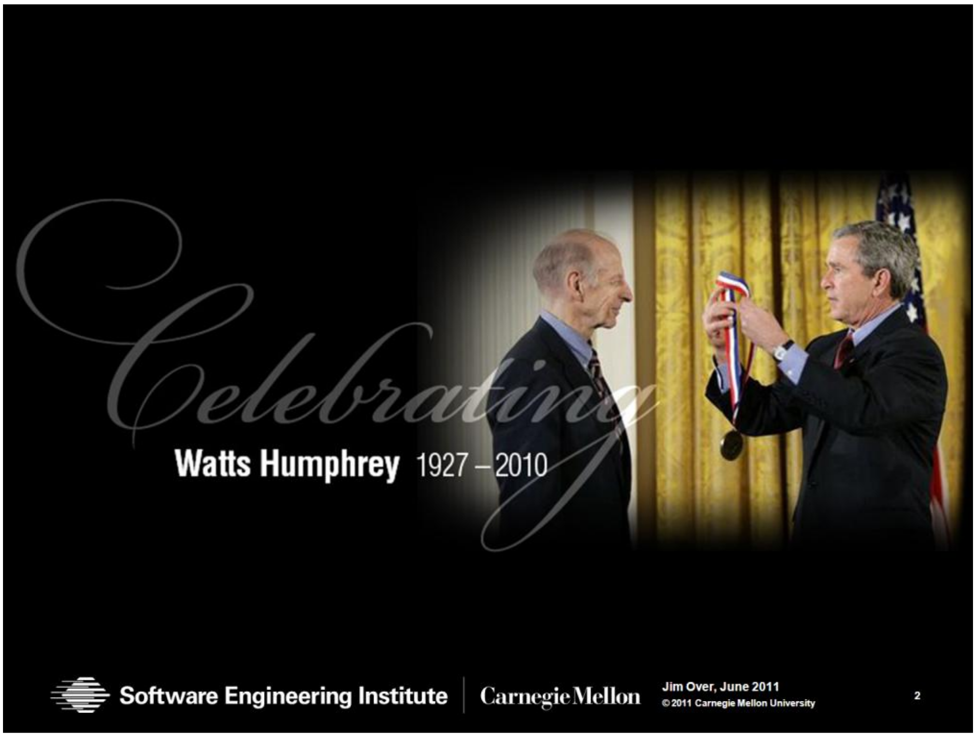


Failure Is Not an Option

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This work is sponsored by the U.S. Department of Defense

Software Engineering Institute | Carnegie Mellon



Celebrating

Watts Humphrey 1927 – 2010

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011

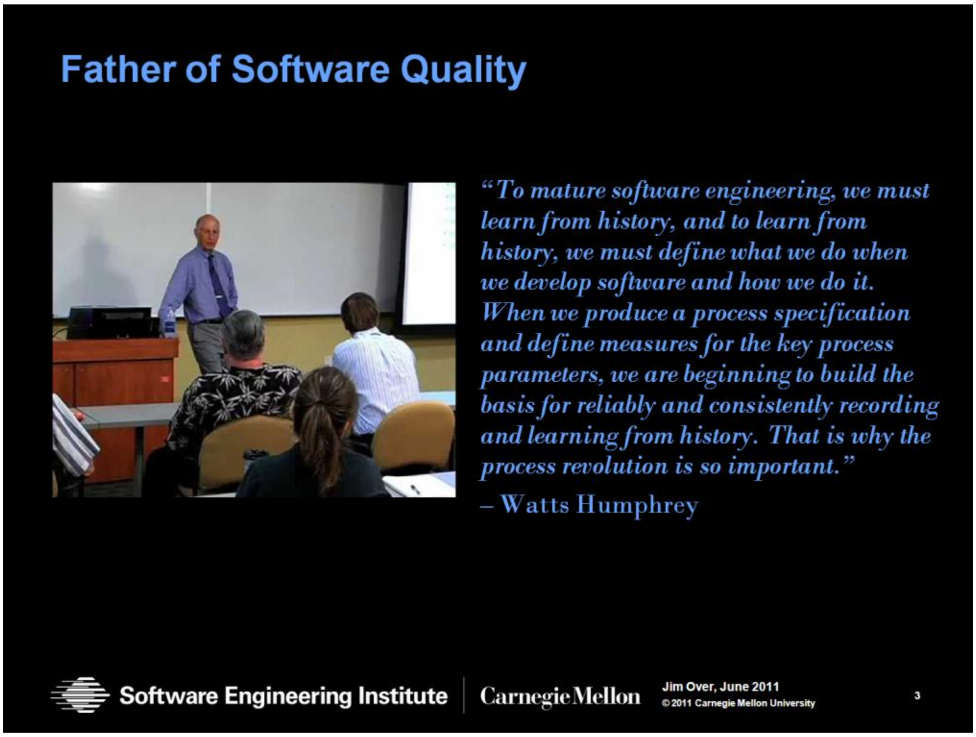
© 2011 Carnegie Mellon University

2

Before beginning the presentation I would like to take a moment to remember Watts Humphrey.

Watts Humphrey joined the SEI in August 1986, at the age of 60, after a long and successful career at IBM.

At an age when many of us are thinking of retirement, Watts elected to start a new career at the SEI because of an outrageous commitment he made to himself to change the world of software engineering by starting a software process revolution. His objective was to change software development from an ad hoc, labor-intensive activity, to a managed, technology supported discipline.



Father of Software Quality

"To mature software engineering, we must learn from history, and to learn from history, we must define what we do when we develop software and how we do it. When we produce a process specification and define measures for the key process parameters, we are beginning to build the basis for reliably and consistently recording and learning from history. That is why the process revolution is so important."

– Watts Humphrey

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

3

Watts became known as the father of software quality for his pioneering work in software process management. He received the United States National Medal of Technology Award from President George Bush for his original five-level model of software process maturity.

His main tenet was that the quality of the software process governed the quality of the product, "to consistently produce reliable, high-quality products to committed plans, the software profession must work to improve the software development process."

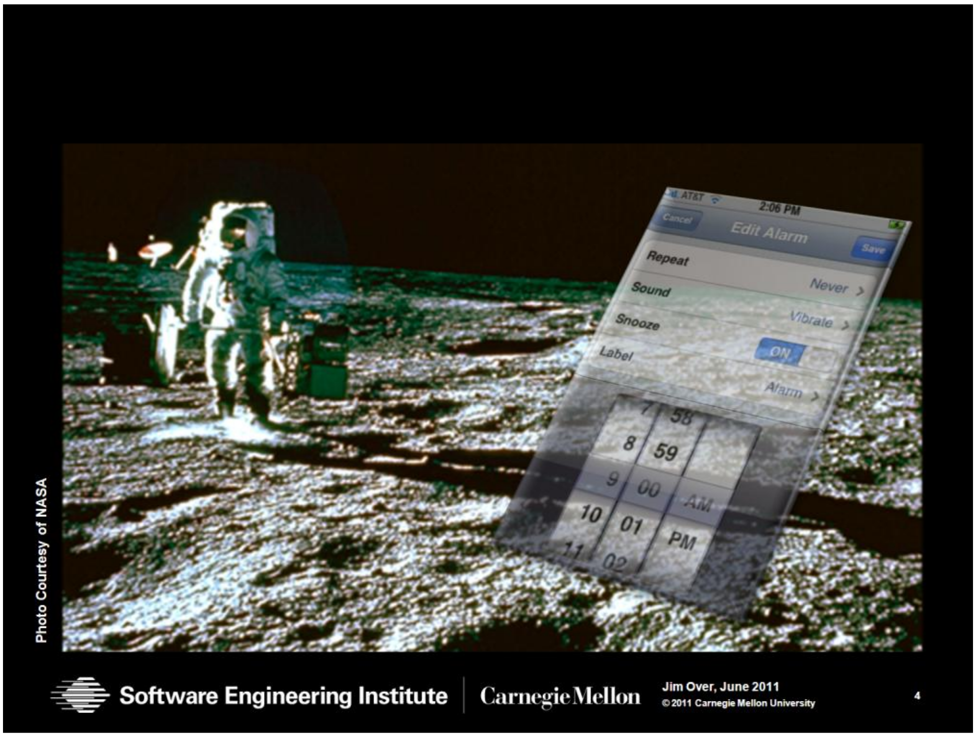


Photo Courtesy of NASA

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011

© 2011 Carnegie Mellon University

4

As the old saying goes, we can put a man on the moon but…we can't build an alarm clock app for the iPhone that knows what to do when a new year begins or how to handle daylight savings time.

Why?

Is software so much more difficult than landing a man on the moon? Is it impossible to build software that works?

Is software such a unique and different kind of engineering that reliability is unattainable or impractical?

With software, failure is not an option… it's a feature.

Software Engineering Institute | Carnegie Mellon

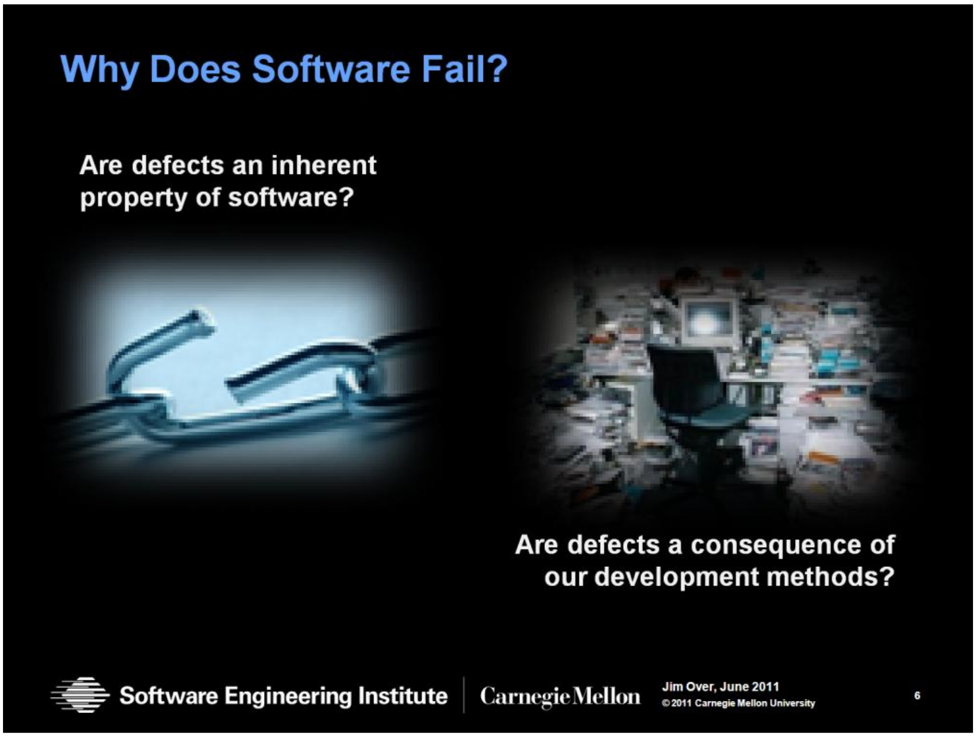
Jim Over, June 2011
© 2011 Carnegie Mellon University

5

With software, failure is not an option, it's a feature that is included in every software program.

Software failures are not restricted to a few classes of non-critical programs. All types of software fail, even critical systems like operating systems, mobile phones, power station control software, and automotive software are full of bugs and fail regularly. Mobile phones, and especially smart phones, often require months of testing by the wireless provider before being approved for use on their network. What makes this even more incredible is that the effective life of these products is only about 18 months. Defects and vulnerabilities in the control systems for power stations are a growing concern. Although the safety record is sill good, software has been responsible for several significant outages and a recent government exercise demonstrated that hackers could take control of a power generating station and destroy the expensive and difficult to replace, one of a kind generators that produce power for the electric grid, leading to very long term outages.

Many applications have more than 5 defects per thousand software instructions, and even widely-used operating systems have more than one defect in every thousand lines of code. Small to medium size applications typically have from 10,000 to 1,000,000 instructions. Operating systems can have close to 100 million lines of code. With 1 to 5 defects per thousand instructions, the software on your computer is incredibly buggy and prone to failure.



Why Does Software Fail?

Are defects an inherent property of software?

Are defects a consequence of our development methods?

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

6

Many believe that software is just buggy. They believe that bug-free code can't be written.

The standard excuses are that requirements change during development, or that schedules are too short to do it right, or that the complexity of software makes it impossible to build a reliable high-quality products. The conventional wisdom is that it is easier and more cost effective to deliver a continuous stream of updates to the software, or just keep fixing it until it works.

Software manufacturers hide behind these excuses to avoid being liable for the products that they produce.

Some software professionals believe that it is our practices that lead to software failure. But even these software developers often lack conviction, and do not practice what they believe. When behind schedule, they cut corners without any sense of the impact on cost or quality.

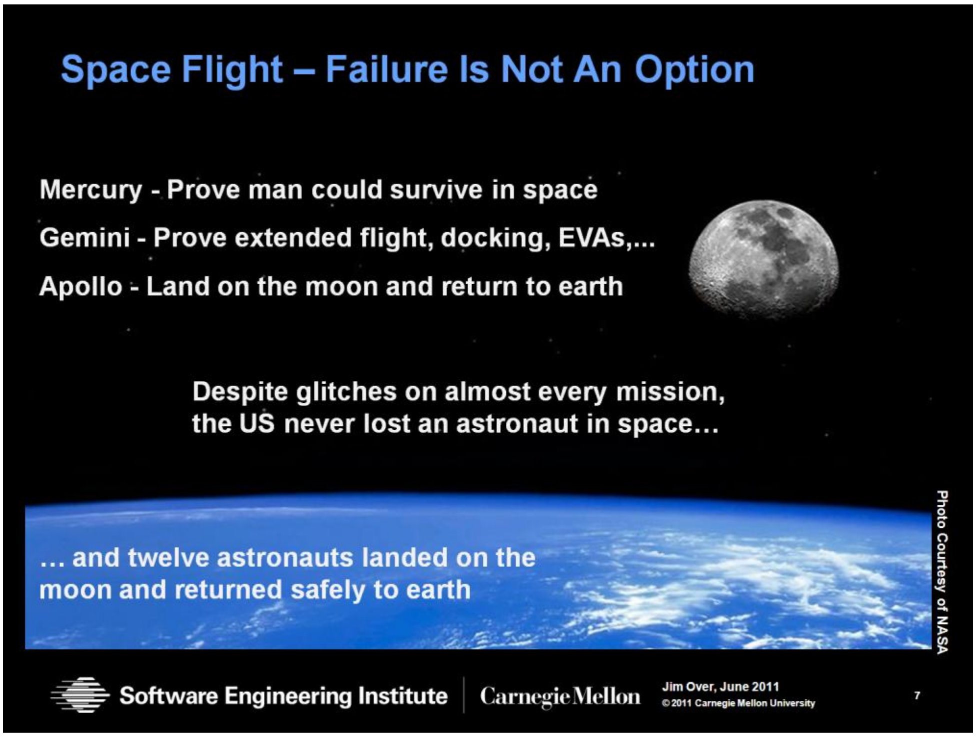
To illustrate, consider formal software inspections. Though proven to be a cost effective method for improving software quality, most software developers do not use formal inspections. Instead they opt for informal and less effective quality management techniques like desk checks, informal reviews, walkthroughs, or pair programming.

Many choose to use these methods only on the "critical code" as if that is the only area where defects occur or count. With these practices many defects slip through testing, making the product unreliable and increasing the risk for the user of the product.

While these techniques should not be discouraged, they are just not as effective from a quality management and an economic, or time-to-market perspective. While dated, the best results are achieved when 100% of the software is inspected.

I believe that software is not inherently buggy. I believe that software fails because of poor development practices. But this is more than a belief, it is a fact. I know that it's possible to build high quality software at lower cost because I have the data to prove it.

But this presentation will not be focused on software practice. Instead the presentation will look at how failure has been avoided in other professions, specifically space flight, medicine, and baseball, with the expectation that maybe we can learn from the experience of others.



Space Flight – Failure Is Not An Option

Mercury - Prove man could survive in space

Gemini - Prove extended flight, docking, EVAs,...

Apollo - Land on the moon and return to earth

Despite glitches on almost every mission,
the US never lost an astronaut in space…

… and twelve astronauts landed on the
moon and returned safely to earth

Photo Courtesy of NASA

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

7

Failure not an option in space flight. Even simple failure can have tragic consequences as the US has learned after the tragic loss of two space shuttle crews.

America's first space program, the program to put a man on the moon, had a better safety record. From the Mercury program in the early 60s through the Apollo program, twelve astronauts landed on the moon and none were lost in space. The only loss of life occurred during static testing of the Apollo spacecraft. Three Astronauts died when a spark from hardware that was known to be faulty ignited the pure oxygen atmosphere within the spacecraft. This accident was caused in part by a failure to follow established safety procedures in the rush to accelerate the Apollo program. It did not speed up the program, but instead cause an enormous delay.

The lesson learned, in space flight, the fastest way to get the job done is to do it right.

Mission Control

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011

© 2011 Carnegie Mellon University

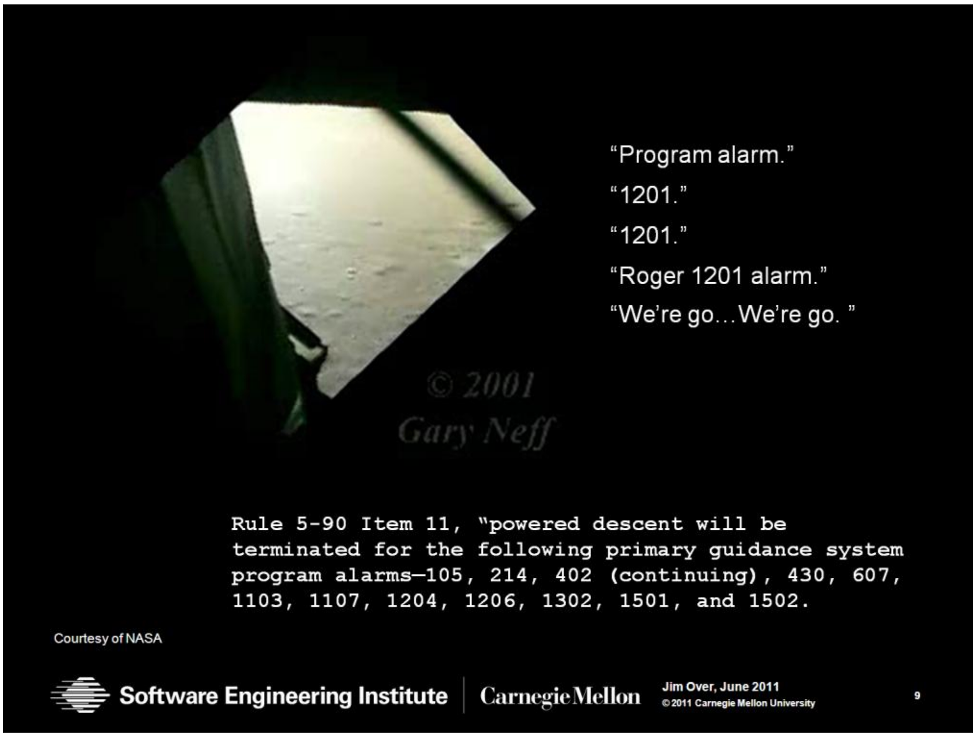
8

Before the space program began, the concept of a mission control group did not exist in flight testing. A test pilot can fly the plane for flight testing. Spacecraft are not like airplanes. A test pilot cannot “fly” a space craft into orbit and return successfully. He or she needs the support of a ground crew and an array of sensors, communications equipment, computers and so forth. This vast network of support was provided and coordinated through mission control.

To many people, Gene Kranz was mission control. He left his job as a military test pilot and joined NASA before the first Mercury flight. His first job was to create the checklists and procedures for the Mercury program. This lead to his leadership position in mission control.

In the movie Apollo 13, Gene is the person that said, “We’ve never lost an astronaut in space and we’re not going to lose one on my watch. Failure is not an option.”

Gene believed that knowledge, skills, discipline, commitment, and teamwork were the keys to success. He made his teams of mission controllers learn everything there was to know about the spacecraft and systems. He developed their skills through hours of practice and simulation. He allowed for human error, but not for a lack of discipline or commitment to the mission. He made the mission controllers document their mission procedures and checklists. He insisted on teamwork. Gene’s approach to mission control and leadership saved many missions.



“Program alarm.”

“1201.”

“1201.”

“Roger 1201 alarm.”

“We’re go…We’re go. ”

Rule 5-90 Item 11, “powered descent will be terminated for the following primary guidance system program alarms—105, 214, 402 (continuing), 430, 607, 1103, 1107, 1204, 1206, 1302, 1501, and 1502.

Courtesy of NASA

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011

9

Many of NASA’s missions, from Alan Shepard’s first Mercury flight to the end of the Apollo program, had one or more glitches. Many of the glitches, though less catastrophic than Apollo 13, were life-threatening situations. Mission controllers, using their knowledge, skills, and discipline, supported by their checklists and procedures, saved these missions and prevented loss of life.

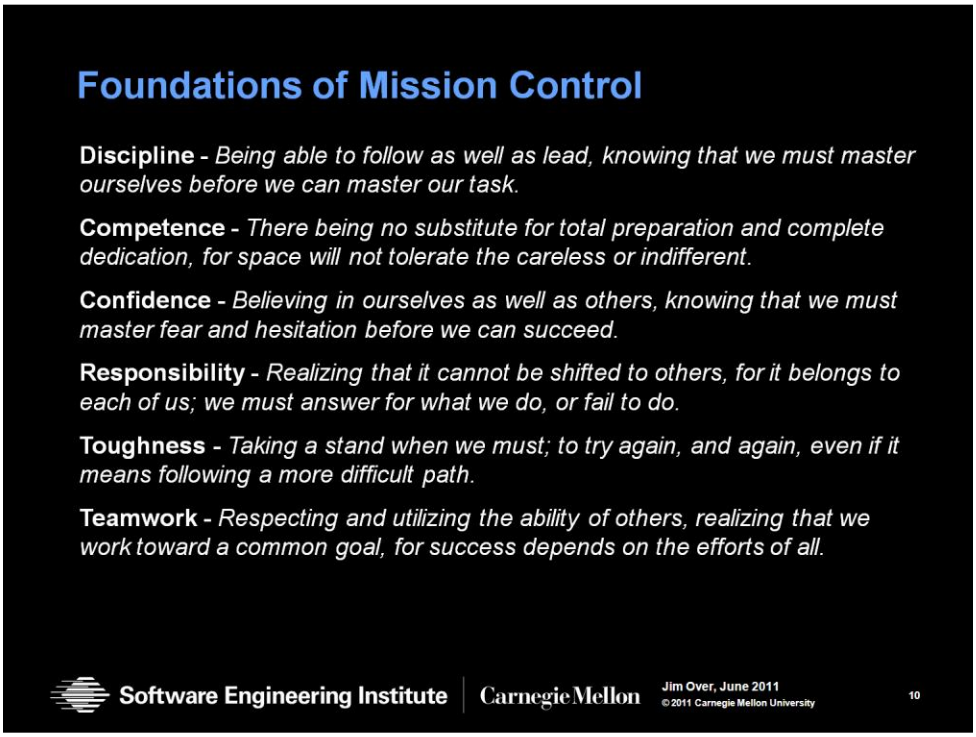
This video from the Apollo 11 moon landing illustrates this point.

The video starts just before lunar touchdown. In the video you will hear members of the crew and mission control deal with a program alarm. The 1201 program alarm indicates a failure in the primary guidance computer as it guides the descent of the lunar module. A failure of the guidance computer could cause the lunar module to crash into the surface of the moon. To avoid the crash, the mission abort sequence would terminate the landing by firing the rockets that are used to lift the lunar module off the surface of the moon and put it back into lunar orbit where it will rendezvous with the command module.

The computers and programming used were very primitive. The guidance computer can report the numeric error code but not a description of the error. The ground computers also report the fault but without an error message. None of the computers or programs were able to analyze the fault and determine what actions to take. Instead, the mission controllers had to perform the analysis, and in a matter of seconds, give the signal to continue or abort.

In the video, 8 seconds elapse from the initial 1201 program alarm code until mission control responds, “we’re go…we’re go”. Within those 8 seconds, the mission controller looks up rule 5-90, which states the alarm codes that require an abort in this phase of the lunar landing and gives the go ahead to continue with the landing. A failure might have led to a loss of life or an unnecessary abort of the Apollo 11 lunar landing.

Today a computer program would have made the same decision in far less than a second. That’s not the point. The issue is that in space flight, failure was not an option. Then and now, our knowledge, skills, discipline, commitment, and teamwork govern the outcome. The controllers took their job seriously. A simple error could cost lives and/or have a damaging effect on the progress of the entire space program. Today that responsibility may rest on the software team that develops the program that makes the critical decision. Do they take their job as seriously as Gene Kranz and his team of mission controllers did?



Foundations of Mission Control

Discipline - *Being able to follow as well as lead, knowing that we must master ourselves before we can master our task.*

Competence - *There being no substitute for total preparation and complete dedication, for space will not tolerate the careless or indifferent.*

Confidence - *Believing in ourselves as well as others, knowing that we must master fear and hesitation before we can succeed.*

Responsibility - *Realizing that it cannot be shifted to others, for it belongs to each of us; we must answer for what we do, or fail to do.*

Toughness - *Taking a stand when we must; to try again, and again, even if it means following a more difficult path.*

Teamwork - *Respecting and utilizing the ability of others, realizing that we work toward a common goal, for success depends on the efforts of all.*

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

10

The Mission Control team followed these principles. Although not designed for software teams, many of these concepts apply. Software engineering is also serious work, and the need for discipline and teamwork are just as important.

Surgery – Failure Is Not An Option

150,000 deaths following surgery in the US every year

Central Line Infections

80,000 people a year

Fatal between 5% and 28% of the time

Source – Gawande, Atul (2010). The Checklist Manifesto: How to get things right

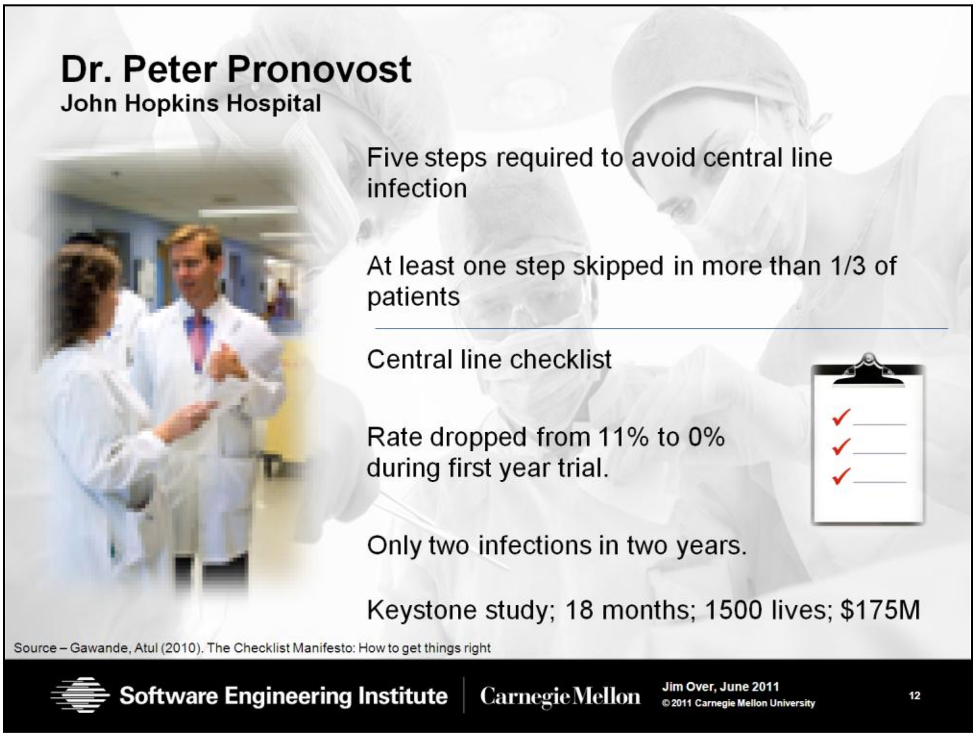
Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

11

Failure also not an option in medicine, especially in surgery, at least that's what you want to believe. In the United States more than 150,000 people die every year from complications that develop after surgery. That's three times the number of people that die from automobile accidents every year. One common complication that can result in death is a central line infection. Though easily prevented, there are about 80,000 central line infections (CLI) per year, resulting in 4,000 fatalities.

These are failures with the potential for grave consequences. What can be done?



Dr. Peter Pronovost

John Hopkins Hospital

Five steps required to avoid central line infection

At least one step skipped in more than 1/3 of patients

Central line checklist

Rate dropped from 11% to 0% during first year trial.

Only two infections in two years.

Keystone study; 18 months; 1500 lives; $175M

Source – Gawande, Atul (2010). The Checklist Manifesto: How to get things right

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
©2011 Carnegie Mellon University

12

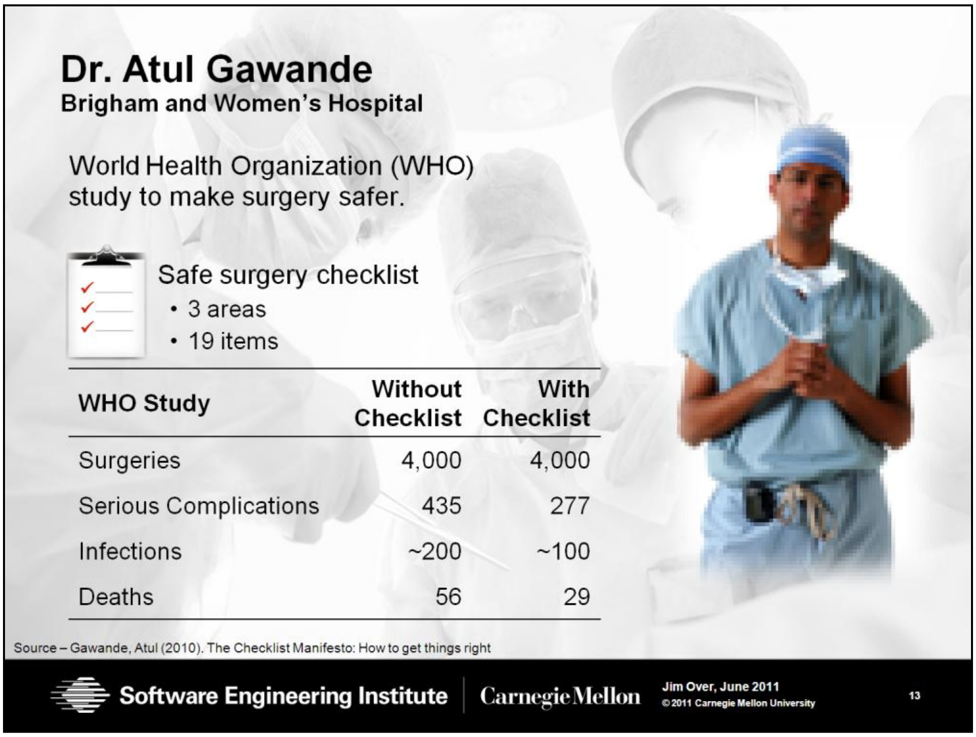
In studies conducted by Dr. Peter Pronovost at John Hopkins, it was found that critical steps required to prevent infections were skipped. When inserting a central line into a patient, doctors must perform five steps to prevent an infection. The study concluded that one or more steps were skipped in more than one-third of the central line insertions.

The results suggest that some of the most knowledgeable, most skilled, and most trusted individuals in our society cannot follow a simple procedure with anymore success than you or I would have if we went grocery shopping without a list.

Why? It’s not stupidity, not a lack of education or training, it’s human nature.

To combat this problem Dr. Pronovost turned to the checklist. The same tool that Gene Kranz used in mission control. Dr. Pononvost created a five step checklist for inserting a central line, and assigned the responsibility for checking that the steps were complete to a nurse, not the doctor. He then applied this method in the ICU for one year. During that year central line infections dropped to zero. The results were so startling that he continued the study for another year. During this entire period only two central line infections occurred.

Based on these results, BCBS Keystone conducted a similar study across the state of Michigan for 18 months. The results were impressive. The CLI checklist saved 1500 lives and $175,000,000, a very impressive result for such a simple solution.



Dr. Atul Gawande

Brigham and Women's Hospital

World Health Organization (WHO) study to make surgery safer.

Safe surgery checklist
- 3 areas
- 19 items

WHO Study	Without Checklist	With Checklist
Surgeries	4,000	4,000
Serious Complications	435	277
Infections	~200	~100
Deaths	56	29

Source – Gawande, Atul (2010). The Checklist Manifesto: How to get things right

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

13

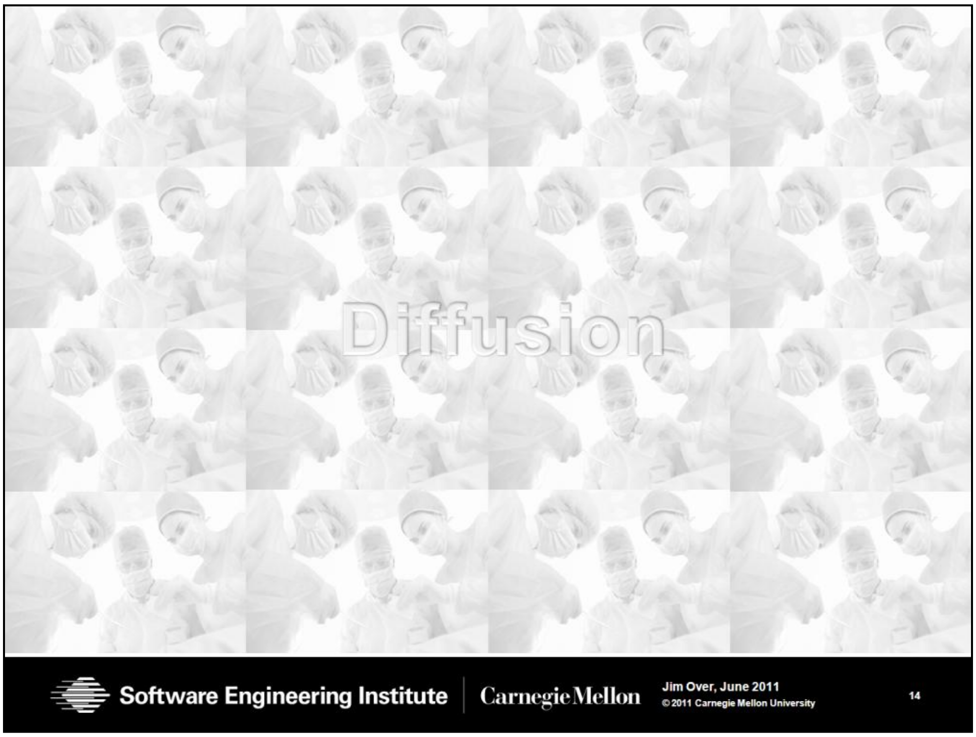
When Dr. Atul Gawande was asked to participate in a World Health Organization program to make surgery safer, he chose to build on the CLI checklist concept and create a safe surgery checklist. But making surgery safer around the world is a much more challenging problem. In many underdeveloped countries the lack of training, lack of basic medicines, and the lack of surgical equipment make surgery less safe.

Believing it would be worth trying, Dr. Gawande, working with others from the safe surgery program, created a safe surgery checklist. They tried to put into a checklist, or process, all the steps necessary to make surgery safe. The initial effort proved to be a failure when Dr. Gawande tried to use it himself. His surgical team found that the list was incomplete and ambiguous making it hard to follow. To fix these issues they produced a more precise and complete checklist. This checklist also failed. It was too complicated to use during surgery when split-second decisions are called for.

Not wanting to give up, Dr. Gawande decided to visit Boeing and discuss his problem with the pilots that develop checklists for Boeing aircraft. From Boeing he learned that the checklist must be short and simple, and only include what the pilot would normally forget. Using these guidelines Dr Gawande started over. The final safe surgery checklist had just nineteen steps in three areas, pre-anesthesia, pre-incision, and close. The steps aren't all about medicine. The first item is introductions. Each member of the team introduces themselves, and describes any special conditions for the team to consider from their perspective. This helps to build the team, identifies special concerns, and improves communication.

How well did it work? Like the CLI checklist the results were hard to believe. Working in eight hospitals around the world a team of analysts complied a baseline for three months. In some 4,000 surgeries in these eight hospitals there were 435 post-surgery complications, including approximately 200 post-surgery infections, and 56 deaths. Using the safe surgery checklist for the same period there were only 277 post-surgery complications, approximately 100 post-surgery infections, and only 29 deaths.

Just like the CLI checklist, the safe surgery checklist was a success.



Diffusion

Software Engineering Institute | Carnegie Mellon

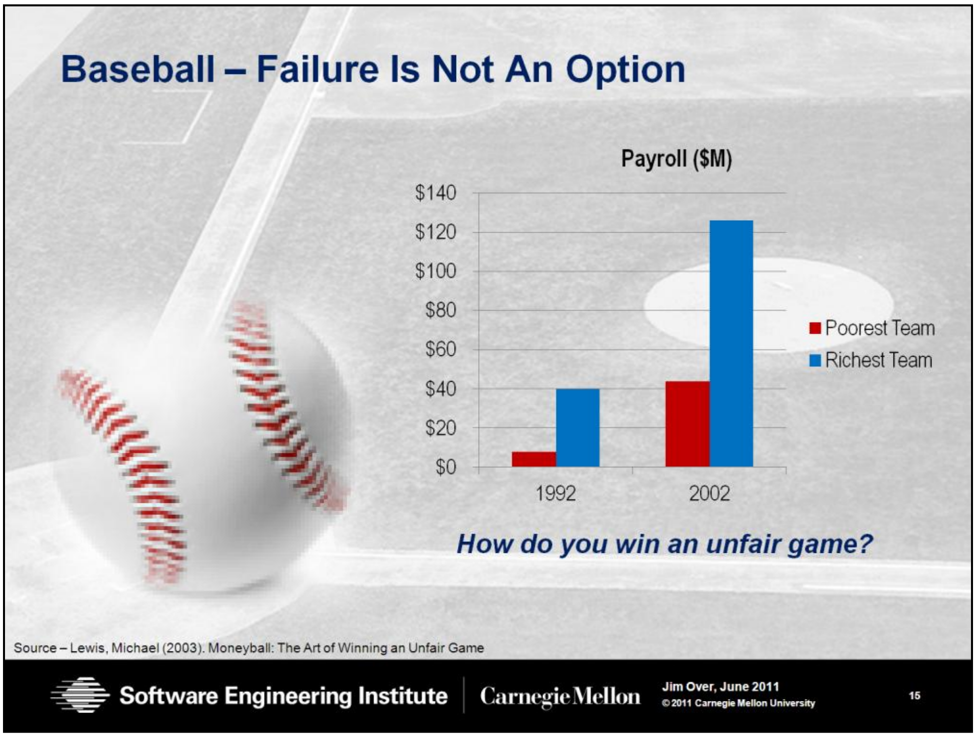
Jim Over, June 2011
© 2011 Carnegie Mellon University

14

But is the safe surgery checklist working?

Dr. Gawande and his colleagues learned that a checklist, like a process, has to be simple and pragmatic or it won't be used. The checklist should address only those items that have the most value and/or that have been shown, through failure, are likely to be forgotten. They learned that introducing checklists into medicine involves culture change. Doctors sounding a bit like software developers, believed that they were infallible. They don't make mistakes, and they don't need checklists. Change management practices were needed.

While still not widely used in surgery, there has been some important progress. From 2009 to 2010 the safe surgery checklist was adopted in 12 countries, including 10% of US hospitals, and more than 2,000 hospitals worldwide.



Baseball – Failure Is Not An Option

How do you win an unfair game?

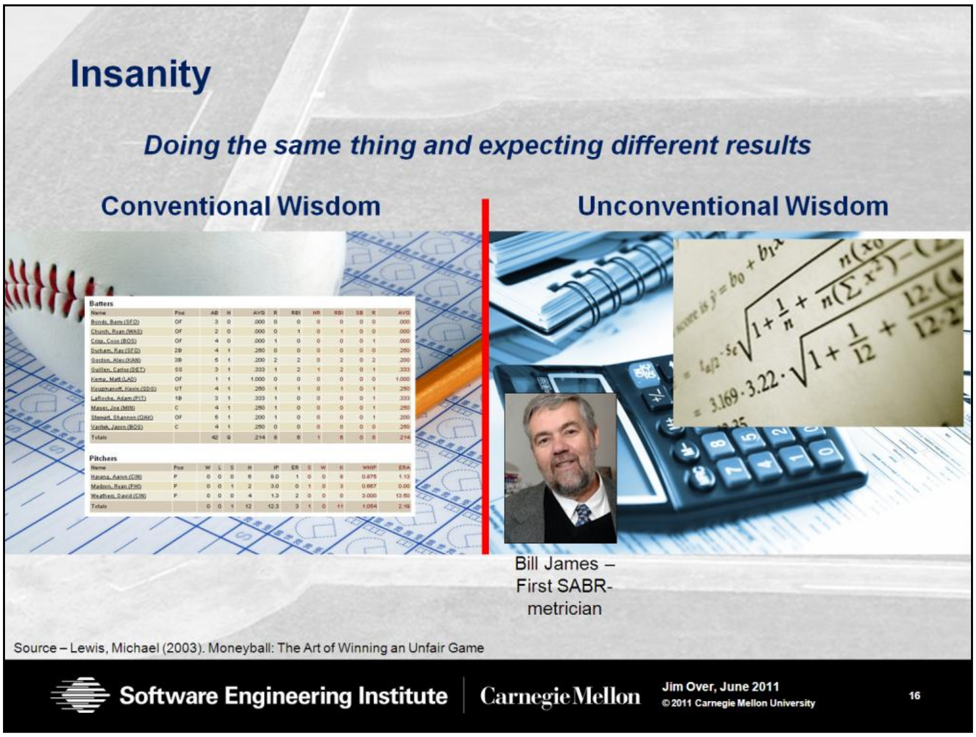
Source – Lewis, Michael (2003). Moneyball: The Art of Winning an Unfair Game

Is failure an option in baseball? There is generally no risk of loss of life, but there is the potential for loss of revenue and potentially the loss of your job.

This is a story about winning. Winning a game that is unfair. In baseball winning is hard because the deck is stacked in favor of the richest teams. They can afford the best players and therefore have a much greater likelihood of winning.

The disparity is substantial, ranging from 3 to 5 times over the ten year period from 1992 to 2002.

So how do you win an unfair game? One team found a way.



Insanity

Doing the same thing and expecting different results

Conventional Wisdom

Unconventional Wisdom

Batters

Name	Pos	AB	H	AVG	R	RBI	HR	RBI	SB	R	AVG
Bonds, Barry (SFO)	OF	3	0	.000	0	0	0	0	0	0	.000
Church, Ryan (WAS)	OF	2	0	.000	0	1	0	1	0	0	.000
Crisp, Coco (BOS)	OF	4	0	.000	1	0	0	0	0	1	.000
Durham, Ray (SFO)	2B	4	1	.250	0	0	0	0	0	0	.250
Gordon, Alex (KAN)	3B	5	1	.200	2	2	0	2	0	2	.200
Guillen, Carlos (DET)	SS	3	1	.333	1	2	1	2	0	1	.333
Kemp, Matt (LAD)	OF	1	1	1.000	0	0	0	0	0	0	1.000
Kouzmanoff, Kevin (SDG)	UT	4	1	.250	1	1	0	1	0	1	.250
LaRoche, Adam (PIT)	1B	3	1	.333	1	0	0	0	0	1	.333
Mauer, Joe (MIN)	C	4	1	.250	1	0	0	0	0	1	.250
Stewart, Shannon (OAK)	OF	5	1	.200	1	0	0	0	0	1	.200
Varitek, Jason (BOS)	C	4	1	.250	0	0	0	0	0	0	.250
Totals		42	9	.214	8	6	1	6	0	8	.214

Pitchers

Name	Pos	W	L	S	H	IP	ER	S	W	K	WHIP	ERA
Harang, Aaron (CIN)	P	0	0	0	6	8.0	1	0	0	8	0.875	1.13
Madson, Ryan (PHI)	P	0	0	1	2	3.0	0	1	0	3	0.667	0.00
Weathers, David (CIN)	P	0	0	0	4	1.3	2	0	0	0	3.000	13.50
Totals		0	0	1	12	12.3	3	1	0	11	1.054	2.19

Bill James – First SABR-metrician

Source – Lewis, Michael (2003). Moneyball: The Art of Winning an Unfair Game

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
©2011 Carnegie Mellon University

16

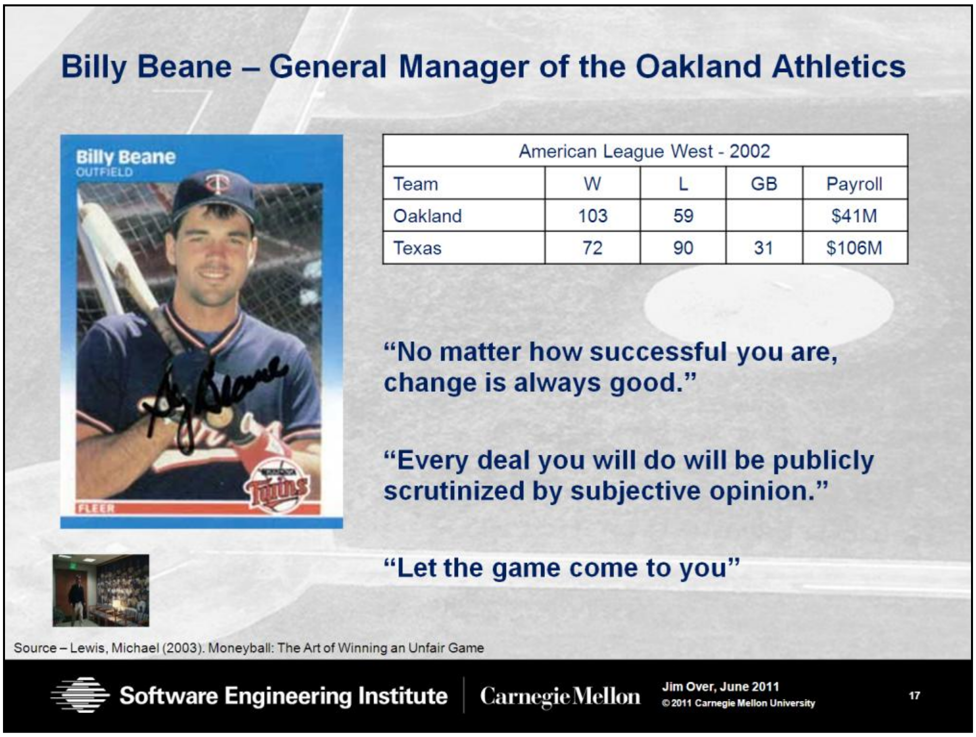
Is baseball an unfair game? Are the best players the players with the highest salaries? Do the best players make the best team?

The conventional wisdom is to pay top salaries and hire the best players. What does best mean? What are the right criteria?

In baseball, the baseball scouts is responsible for finding and recruiting the best players. They look for players with strength and speed in hitting and/or pitching, the hitter that can hit the ball the farthest or the pitcher that can throw the 95+ mile per hour fast ball. They also tend to favor players that have the "look" of a major leaguer. These factors determine what a player is worth.

Are these criteria the most important? Do they lead to a winning season?

In the 1980s, Bill James, a baseball fan with an interest in statistics, began a decade long examination of the criteria most commonly used to evaluate players and determined that they were not necessarily predictive of winning seasons. His work led to a "movement" where like minded individuals such as economists, statisticians, and others began to develop new, more predictive statistics. Using data from past seasons they were able to show that these statistics were more predictive of the outcome than those in common use. With a sense of triumph they approached major league baseball teams and were surprised to learn that no one was interested in this unconventional wisdom.


Billy Beane
OUTFIELD
FLEER
Twins

Billy Beane – General Manager of the Oakland Athletics

American League West - 2002				
Team	W	L	GB	Payroll
Oakland	103	59		$41M
Texas	72	90	31	$106M

"No matter how successful you are, change is always good."

"Every deal you will do will be publicly scrutinized by subjective opinion."

"Let the game come to you"

Source – Lewis, Michael (2003). Moneyball: The Art of Winning an Unfair Game

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

17

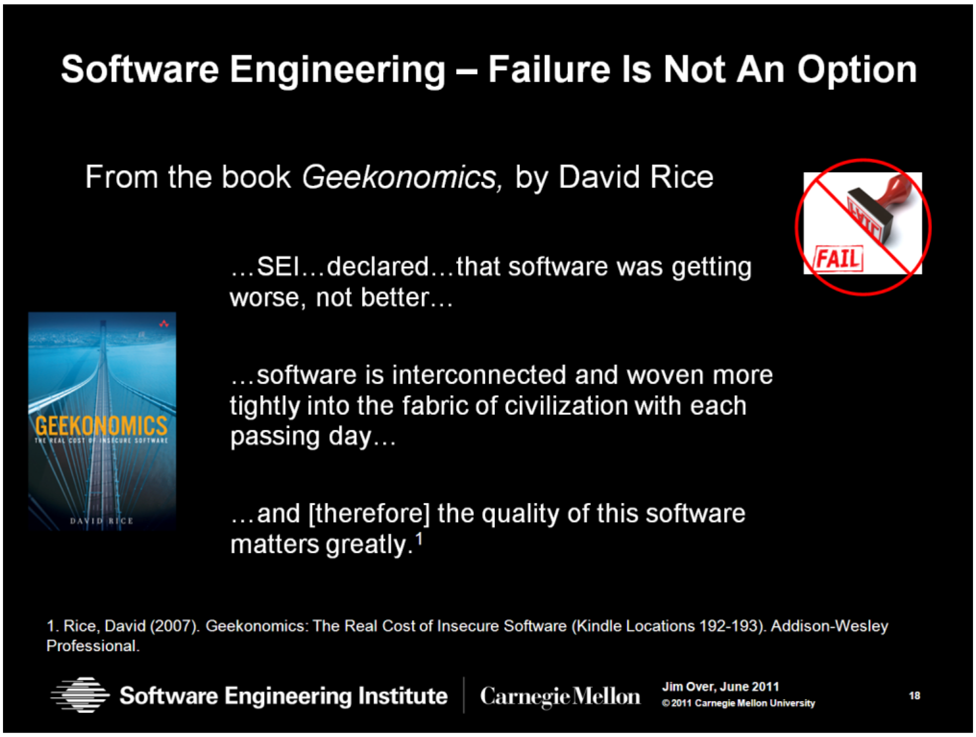
In 1998, an ex-ball player, Billy Bean, was hired as the General Manager of the Oakland Athletics. Billy had been one of those players that was recruited out of high school for his speed and strength, and because he had the "looks" of a major league player. But he never matured as a player. As his baseball career slumped, he watched less talented players have greater success. Frustrated after years of failure, he quit baseball.

Soon after Billy Bean became the General Manager of the Oakland Athletics, the new owners cut the team's payroll to reduce losses. This event, together with Billy's failure as a ball player, and Bill James new statistics created the opportunity needed to overcome conventional wisdom.

The advantage of Bill James' statistics is that they were not understood by professional baseball scouts and so players with these more important characteristics were under-valued. Billy Bean knew from his own experience that those players that were valued by the scouts did not necessarily lead to a winning season. Billy Bean thought that by using Bill James statistics, he might build teams using players that nobody else wanted, and have winning seasons with a lower payroll.

For the next several years, from 1999 to 2003 he did just that. The Oakland Athletics were able to be 1st or 2nd in the American League West, even though they had the lowest salary in the league. He accomplished this using data and by building winning teams, teams whose collective strengths compensated for their weaknesses.

He often said, "Let the game come to you," meaning, don't try to force a win. Trying to force a win can involve unnecessary risk that will change the basis for the winning season. In other words, there is no need to act in an ad hoc, reactive way to try and win the game. Doing so will decrease your odds of winning. Instead, let the game come to you.



Software Engineering – Failure Is Not An Option

From the book *Geekonomics,* by David Rice

…SEI…declared…that software was getting worse, not better…

…software is interconnected and woven more tightly into the fabric of civilization with each passing day…

…and [therefore] the quality of this software matters greatly.[1]

[1] Rice, David (2007). Geekonomics: The Real Cost of Insecure Software (Kindle Locations 192-193). Addison-Wesley Professional.

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

18

Failure is certainly not an option anymore in software engineering. Software has become the infrastructure on which modern society is based, but unfortunately software quality continues to decline.

The degree of interconnectivity provided by the Internet means that software quality matters even more than in the past. Today the software glitch is not just annoying, it could be exploited to create a network attack, steal an identity, access private data, or create a financial opportunity for the hacker.

For the software profession, failure is no longer an option, and we need to do something now.

Discipline
Trust
Teamwork
Process
Measurement
Facts and Data
Quality

Discipline
Trust
Teamwork
Process
Measurement
Facts and Data
Quality

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

19

High quality requires disciplined teams.
Process governs the quality that can be achieved.
Complex work must be guided or steps will be missed.
You must measure to manage and learn.
Trust in the data; let the game come to you.
Utility can be achieved at lower cost if you understand the game.

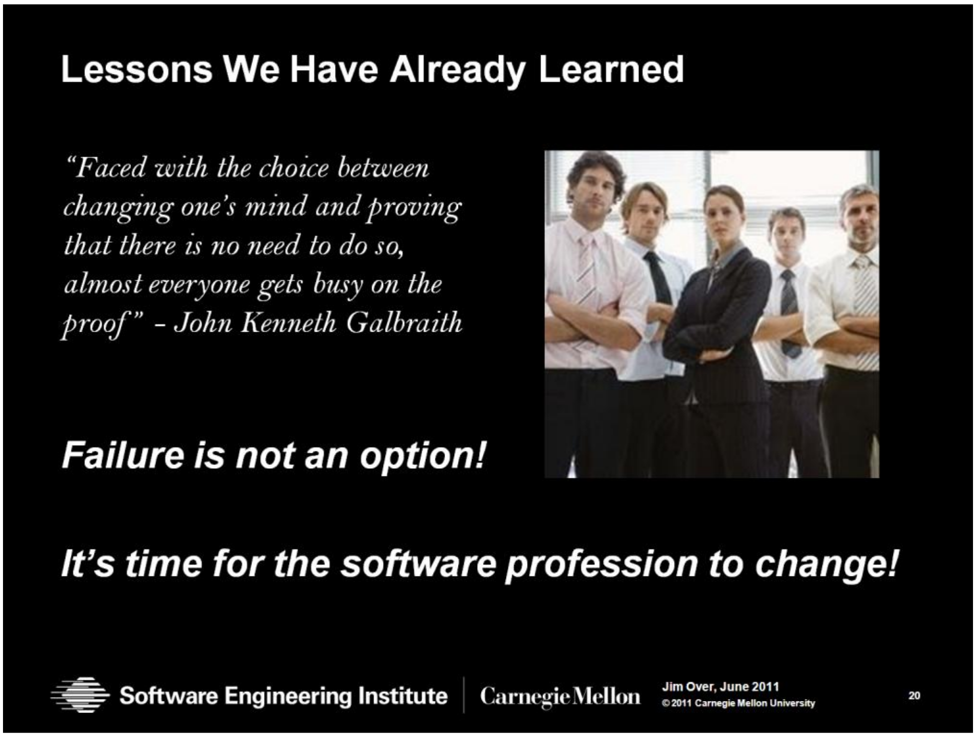
Are the examples of these other professions of any practical value for software engineering?

Can we learn anything from space flight, medicine, or baseball?

Of course we can.

In all of the examples, failure was not considered to be an option. That is, the quality of the results was paramount. Whether the focus was a successful space mission, a successful surgery, or a winning baseball season, the outcome mattered.

Recognizing that complex work must be guided, these professions used checklists, procedure, and process to ensure that they achieved their best performance. In every setting, the teams were disciplined. They used, measured, and refined these checklists, procedures, and processes. They used data not conventional wisdom to manage and learn. Having learned, they had the patience to trust the data and “let the game come to them.”



Lessons We Have Already Learned

"Faced with the choice between changing one's mind and proving that there is no need to do so, almost everyone gets busy on the proof" – John Kenneth Galbraith

Failure is not an option!

It's time for the software profession to change!

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011

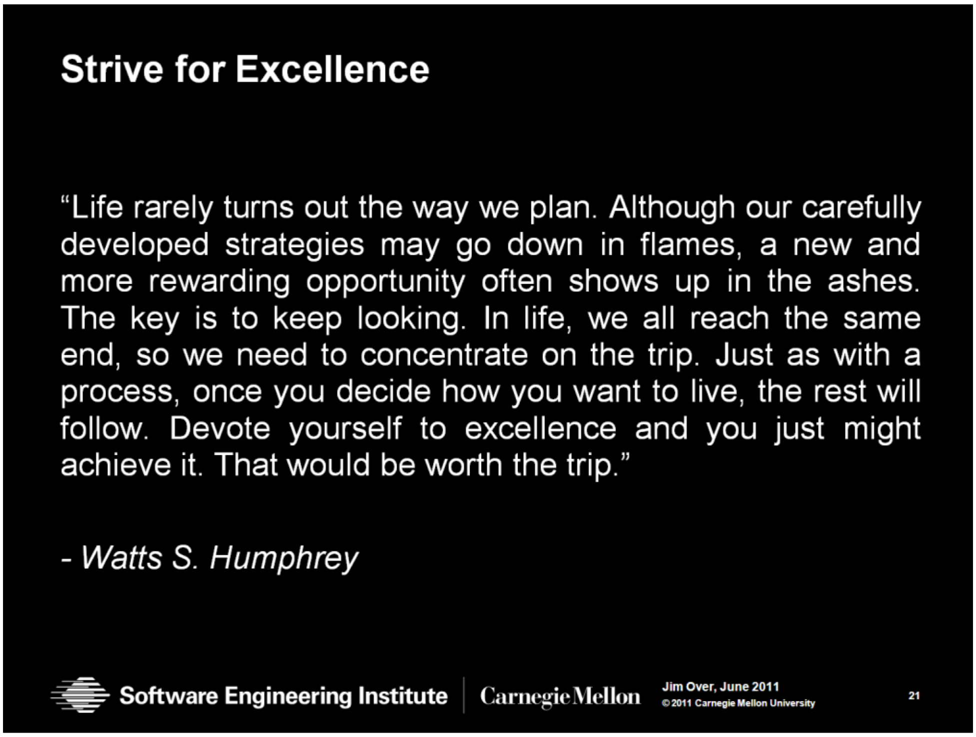
© 2011 Carnegie Mellon University

20

But we know that already; so why aren't we doing something about it?

John Kenneth Galbraith said, "Faced with the choice between changing one's mind and proving that there is no need to do so, almost everyone gets busy on the proof."

Certainly the time has come for us to stop working on the proof and get busy on the change. We must succeed. Failure is not an option!



Strive for Excellence

"Life rarely turns out the way we plan. Although our carefully developed strategies may go down in flames, a new and more rewarding opportunity often shows up in the ashes. The key is to keep looking. In life, we all reach the same end, so we need to concentrate on the trip. Just as with a process, once you decide how you want to live, the rest will follow. Devote yourself to excellence and you just might achieve it. That would be worth the trip."

- Watts S. Humphrey

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

21

In the last paragraph of the last chapter of Watts Humphrey's book "A Discipline for Software Engineering," Watts wrote that to make the most of our lives, we should all strive for excellence, and we just might achieve it. It is time for our profession to take his closing remarks seriously. Society demands it.

Software Engineering Institute | Carnegie Mellon

Jim Over, June 2011
© 2011 Carnegie Mellon University

22